IDIAP RESEARCH REPORT

KERNEL BASED TEXT-INDEPENDNENT SPEAKER VERIFICATION

Johnny Mariéthoz Samy Bengio Yves Grandvalet

Idiap-RR-68-2008

SEPTEMBER 2008

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Kernel Based Text-Independnent Speaker Verification

Johnny Mariéthoz[1] Samy Bengio[2] Yves Grandvalet[3]

September 30, 2008

[1]Idiap Research Institute, CP 592, 1920 Martigny, Switzerland and Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, marietho@idiap.ch
[2]Google Inc, Mountain View, CA, USA, bengio@google.com
[3]Heudiasyc, CNRS/UTC, 60205 Compiègne cedex, France grandval@utc.fr

Contents

Chapter 1

Kernel-Based Text-Independent Speaker Verification

The goal of a person authentication system is to authenticate the claimed identity of a user. When this authentication is based on the voice of the user, without respect of what the user exactly said, the system is called a text-independent speaker verification system.

Speaker verification systems are increasingly often used to secure personal information, particularly for mobile phone based applications. Furthermore, text-independent versions of speaker verification systems are the most used for their simplicity, as they do not require complex speech recognition modules. The most common approach to this task is based on Gaussian Mixture Models (GMMs) [RQD00], which do not take into account any temporal information. GMMs have been intensively used thanks to their good performance, especially with the use of the Maximum A Posteriori (MAP) [GL94] adaptation algorithm. This approach is based on the density estimation of an impostor data distribution, followed by its adaptation to a specific client data set. Note that the estimation of these densities is not the final goal of speaker verification systems, which is rather to discriminate the client and impostor classes; hence discriminative approaches might appear good candidates for this task as well.

As a matter of fact, Support Vector Machine (SVM) based systems have been the subject of several recent publications in the speaker verification community, in which they obtain similar to or even better performance than GMMs on several text-independent speaker verification tasks. In order to use SVMs or any other discriminant approaches for speaker verification, several modifications from the classical techniques need to be performed. The purpose of this chapter is to present an overview of discriminant approaches that have been used successfully for the task of text-independent speaker verification, to analyze their difference and their similarities with each other and with classical generative

approaches based on GMMs. An open-source version of the C++ source code used to performed all experiments described in this chapter can be found at `http://speaker.abracadoudou.com`.

1.1 Introduction

Person authentication systems are in general designed in order to let genuine clients access a given service while forbidding it to impostors. This can be seen as a 2-class classification problem suitable for machine learning approaches.

A number of specificities make speaker verification different from a standard binary classification problem. First, the input data are sentences whose lengths depend on its phonetic content and the speaking rate of the underlying speaker.

Second, only few client training examples are available: in most real application, it is not possible to ask a client to speak during several hours or days in order to capture the entire variability of his/her voice. There are typically between one and three utterances for each client.

Third, the impostor distribution is not known and even not well defined: we have no idea of what an impostor is in a "real" application. In order to simulate impostor accesses, one usually considers other speakers in the database. This ignorance is somewhat remedied by evaluating the models with impostor identities that are not available when creating the models. This incidentally means that plenty of impostor accesses are usually available, often more than 1000 times the number of client accesses, which makes the problem highly unbalanced.

The distribution of impostors being only loosely defined, the prior probability of each class is unknown, and the cost of each type of error is usually not known beforehand. Thus, one usually selects a model that gives reasonable performance for several possible cost trade-offs.

Finally, the recording conditions change over time. The speaker can be located in several kinds of places: office, street, train station, etc. The device used to perform the authentication can also change between authentication attempts: land line phone, mobile phone, laptop microphone, etc.

That being said, the problem of accepting or rejecting someone's identity claim can be formally stated as a binary classification task. Let $\mathcal{S}$ be a set of clients and $s_i \in \mathcal{S}$ be the i^{th} client of that set. We look for a discriminant function $f(\cdot; \boldsymbol{\vartheta}_i)$ and a decision threshold Δ such that

$$f(\bar{\mathbf{x}}; \boldsymbol{\vartheta}_i) > \Delta \ , \tag{1.1}$$

if and only if sentence $\bar{\mathbf{x}}$ was pronounced by speaker s_i.

The parameters $\boldsymbol{\vartheta}_i$ are typically determined by optimizing an empirical criterion computed on a set of L_i sentences, either called the training or the learning set $\mathcal{L}_i = \{(\bar{\mathbf{x}}_l, y_l)\}_{l=1}^{L_i}$, where $\bar{\mathbf{x}}_l \in \mathbb{R}^{d \times T_l}$ is an input waveform sequence encoded as T_l d-dimensional frames, and $y_l \in \{-1, 1\}$ is the corresponding target, where 1 stands for for a true client sequence and -1 for an impostor access. The search space is defined as the set of functions $f : \mathbb{R}^{d \times T_l} \mapsto \mathbb{R}$ parameterized by $\boldsymbol{\vartheta}_i$, and

$\boldsymbol{\vartheta}_i$ is identified by minimizing the mean loss on the training set, where the loss $\ell(\cdot)$ returns low values when $f(\bar{\mathbf{x}}; \boldsymbol{\vartheta}_i)$ is near y and high values otherwise:

$$\boldsymbol{\vartheta}_i = \arg\min_{\boldsymbol{\theta}} \sum_{(\bar{\mathbf{x}}, y) \in \mathcal{L}_i} \ell(f(\bar{\mathbf{x}}; \boldsymbol{\theta}), y) \ .$$

Note that the overall goal is not to obtain zero error on $\mathcal{L}_i$ but rather on unseen examples drawn from the same probability distribution. This objective is monitored by measuring the classification performance on an independent test set $\mathcal{T}_i$, in order to provide an unbiased estimate of performance on the population.

A standard taxonomy of machine learning algorithms sets apart discriminant models, that directly estimate the function $f(\cdot; \boldsymbol{\vartheta}_i)$, from generative models, where $f(\cdot; \boldsymbol{\vartheta}_i)$ is defined through the estimation of the conditional distribution of sequences knowing the speaker. We briefly present hereafter the classical generative approach that encompasses the very popular Gaussian Mixture Model (GMM), which will provide a baseline in the experimental section. All the other methods presented in this chapter are kernel-based systems that belong to the discriminative approach.

1.2 Generative Approaches

The state-of-the-art generative approaches for speaker verification use atypical models in the sense that they do not model the joint distribution of inputs and outputs. This is due to the fact that we have no clue of what the prior probability of having client s_i speaking should be, since the distribution of impostors is only loosely defined and the proportion of client accesses in the training set may not be representative of the proportion in future accesses. Although the model is not complete, a decision function is computed using the rationale described below.

1.2.1 Rationale

The system has to decide whether a sentence $\bar{\mathbf{x}}$ was pronounced by speaker s_i or by any other person s_0. It should accept a claimed speaker as a *client* if and only if:

$$P(s_i|\bar{\mathbf{x}}) > \alpha_i P(s_0|\bar{\mathbf{x}}) \ , \tag{1.2}$$

where α_i is a trade-off parameter that accounts for the loss of false acceptance of an impostor access versus false rejection of a genuine client access.

Using Bayes theorem, we rewrite (1.2) as follows:

$$\frac{p(\bar{\mathbf{x}}|s_i)}{p(\bar{\mathbf{x}}|s_0)} > \alpha_i \frac{P(s_0)}{P(s_i)} = \Delta_i = \Delta \ , \tag{1.3}$$

where Δ_i is proportional to the ratio of the prior probabilities of being or not being the client. This ratio being unknown, Δ_i is replaced by a client inde-

pendent decision threshold Δ. This corresponds to having different (unknown) settings for the trade-off parameters α_i.

The left ratio in (1.3) plays the role of $f(\bar{\mathbf{x}}; \boldsymbol{\vartheta}_i)$ in (1.1), where the set of parameters $\boldsymbol{\vartheta}_i$ is decomposed as follows:

$$f(\bar{\mathbf{x}}; \boldsymbol{\vartheta}_i) = \frac{p(\bar{\mathbf{x}}|s_i, \boldsymbol{\theta}_i)}{p(\bar{\mathbf{x}}|s_0, \boldsymbol{\theta}_0)} \ ,$$

with $\boldsymbol{\vartheta}_i = \{\boldsymbol{\theta}_i, \boldsymbol{\theta}_0\}$. The loss function used to estimate $\boldsymbol{\theta}_0$ is the negative log-likelihood

$$\boldsymbol{\theta}_0 = \arg\min_{\boldsymbol{\theta}} \sum_{(\bar{\mathbf{x}}, y) \in \mathcal{L}_i^-} -\log p(\bar{\mathbf{x}}|s_0, \boldsymbol{\theta}) \ ,$$

where $\mathcal{L}_i^-$ is the subset of pairs $(\bar{\mathbf{x}}, y)$ in the learning set $\mathcal{L}_i$ for which $y = -1$. As generally few positive examples are available, the loss function used to estimate $\boldsymbol{\theta}_i$ is based on a Maximum A Posteriori (MAP) adaptation scheme [GL94] and can be written as follows:

$$\boldsymbol{\theta}_i = \arg\min_{\boldsymbol{\theta}} \sum_{(\bar{\mathbf{x}}, y) \in \mathcal{L}_i^+} -\log\Big(p(\bar{\mathbf{x}}|s_i, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \Big)$$

where $\mathcal{L}_i^+$ is the subset of pairs $(\bar{\mathbf{x}}, y)$ in $\mathcal{L}_i$ for which $y = 1$. This MAP approach puts some prior on $\boldsymbol{\theta}$ to constrain these parameters to some reasonable values. In practice, they are constrained to be near $\boldsymbol{\theta}_0$, which represents reasonable parameters for any unknown person. See for instance [RQD00] for a practical implementation.

1.2.2 Gaussian Mixture Models

State-of-the-art systems compute the density of a sentence $\bar{\mathbf{x}}$ by a rough estimate that assumes independence of the T frames that encode $\bar{\mathbf{x}}$. The density of the frames themselves is assumed to be independent of the sequence length, and is estimated by a Gaussian Mixture Model (GMM) with diagonal covariance matrices, as follows:

$$\begin{aligned} p(\bar{\mathbf{x}}|s,\boldsymbol{\theta}) &= P(T)\, p(\bar{\mathbf{x}}|T, s,\boldsymbol{\theta}) \\ &= P(T) \prod_{t=1}^{T} p(\mathbf{x}^t|T, s,\boldsymbol{\theta}) \\ &= P(T) \prod_{t=1}^{T} p(\mathbf{x}^t|s,\boldsymbol{\theta}) \\ &= P(T) \prod_{t=1}^{T} \sum_{m=1}^{M} \pi_m \mathcal{N}(\mathbf{x}^t|\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m) \ , \end{aligned} \tag{1.4}$$

where $P(T)$ is the probability distribution [1] of the length of sequence $\bar{\mathbf{x}}$, $\mathbf{x}^t$ is the t^{th} frame of $\bar{\mathbf{x}}$, and M is the number of mixture components. The parameters $\boldsymbol{\theta}$ comprise the means $\{\boldsymbol{\mu}_m\}_{m=1}^M$, standard deviations $\{\boldsymbol{\sigma}_m\}_{m=1}^M$, and mixing weights $\{\pi_m\}_{m=1}^M$ for all Gaussian components. The Gaussian density is defined as follows:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-2}(\mathbf{x}-\boldsymbol{\mu})\right) \ ,$$

where d is the dimension of $\mathbf{x}$, $\boldsymbol{\Sigma}$ is the diagonal matrix with diagonal elements $\Sigma_{ii} = \sigma_i$, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

As stated in the previous section, we first train an impostor model $p(\bar{\mathbf{x}}|s_0,\boldsymbol{\theta}_0)$, called *world* or *universal background model* when it is common to all speakers s_i. For this purpose, we use the Expectation-Maximization (EM) algorithm to maximize the likelihood of the negative examples in the training set. Note that in order to obtain state-of-the-art performance, the variances of all Gaussian components are constrained to be higher than some threshold, normally selected on a separate development set. This process, often called *variance flooring* [MKLB98], can be seen as a way to control the capacity of the overall model.

For each client s_i, we use a variant of MAP adaptation [RQD00] to estimate a client model $p(\bar{\mathbf{x}}|s_i,\boldsymbol{\theta}_i)$ that only departs partly from the world model $p(\bar{\mathbf{x}}|s_0,\boldsymbol{\theta}_0)$. In this setting, only the mean parameters of the world model are adapted to each client, using the following update rule:

$$\boldsymbol{\mu}_m^i = \tau_{i,m}\,\widehat{\boldsymbol{\mu}}_m^i + (1-\tau_{i,m})\boldsymbol{\mu}_m^0 \ ,$$

where $\boldsymbol{\mu}_m^0$ is the vector of means of Gaussian m of the world model, $\widehat{\boldsymbol{\mu}}_m^i$ is the corresponding vector estimated by maximum likelihood on the sequences available for client s_i, and τ_i is the adaptation factor that represents the faith we have in the client data. The latter is defined as follows [RQD00]:

$$\text{where } \tau_{i,m} = \frac{n_{i,m}}{n_{i,m}+r} \tag{1.5}$$

where $n_{i,m}$ is the effective number of frames used to compute $\widehat{\boldsymbol{\mu}}_m^i$, that is, the sum of memberships to component m for all the frames of the training sequence(s) uttered by client s_i (see Section 1.5.2 for details). The MAP relevant factor r is chosen by cross-validation.

Finally, when all GMMs have been estimated, one can instantiate (1.3) to take a decision for a given access as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\log\frac{\sum_{m=1}^M \pi_m\,\mathcal{N}(\mathbf{x}^t;\boldsymbol{\mu}_m^i,\boldsymbol{\sigma}_m)}{\sum_{m=1}^M \pi_m\,\mathcal{N}(\mathbf{x}^t;\boldsymbol{\mu}_m^0,\boldsymbol{\sigma}_m)} > \log\Delta \ ,$$

[1] Under the reasonable assumption that the distributions of sentence length are identical for each speaker, this distribution does not play any discriminating role and can be left unspecified.

where $\boldsymbol{\theta}_0 = \{\boldsymbol{\mu}_m^0, \boldsymbol{\sigma}_m, \pi_m\}_{m=1}^M$ are the GMM parameters for the world model, and $\boldsymbol{\theta}_i = \{\boldsymbol{\mu}_m^i, \boldsymbol{\sigma}_m, \pi_m\}_{m=1}^M$ are the GMM parameters for the client model. Note that $\frac{1}{T}$ does not follow from (1.3) and is an empirical normalization factor added to yield a threshold Δ that is independent of the length of the sentence.

1.3 Discriminative Approaches

Support Vector Machines (SVMs) [Vap00] are now a standard tool in numerous applications of machine learning, such as in text or vision [Joa02, PV98]. While GMM is the mainstream generative model in speaker verification, SVMs are prevailing in the discriminative approach. This section provides a basic description of SVMs that introduces the kernel trick that relates feature expansions to kernels, on which will focus in Section 1.5.

1.3.1 Support Vector Machines

In the context of binary classification problems, the SVM decision function is defined by the sign of

$$f(\mathbf{x}; \boldsymbol{\vartheta}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b \ , \tag{1.6}$$

where $\mathbf{x}$ is the current example, $\boldsymbol{\vartheta} = \{\mathbf{w}, b\}$ are the model parameters and $\Phi(\cdot)$ is a mapping, chosen "a priori", that associates a possibly high dimensional feature to each input data.

The SVM training problem consists in solving the following problem:

$$\left\{ \begin{array}{ll} (\mathbf{w}^*, b^*) = \arg\min\limits_{(\mathbf{w},b)} \dfrac{1}{2}\|\mathbf{w}\|^2 + C\sum\limits_{l=1}^{L} \xi_l & \\ \text{s.t.} \qquad y_l(\mathbf{w} \cdot \mathbf{x}_l + b) \geq 1 - \xi_l \quad \forall_l & \\ \qquad\quad \xi_l \geq 0 \quad \forall_l \ , & \end{array} \right. \tag{1.7}$$

where L is the number of training examples, the target class label $y_l \in \{-1, 1\}$ corresponds to $\mathbf{x}_l$, and C is a hyper-parameter that trades off the minimization of classification error (upper-bounded by ξ_l) and the maximization of the margin, which provides generalization guarantees [Vap00].

Solving (1.7) leads to a discriminant function expressed as a linear combination of training examples in the feature space $\Phi(\cdot)$. We can thus rewrite (1.6) as follows:

$$f(\mathbf{x}; \boldsymbol{\vartheta}) = \sum_{l=1}^{L} \alpha_l y_l \, \Phi(\mathbf{x}_l) \cdot \Phi(\mathbf{x}) + b \ ,$$

where most training examples do not enter this combination ($\alpha_l = 0$); the training examples for which $\alpha_l \neq 0$ are called *support vectors*.

As the feature mapping $\Phi(\cdot)$ only appears in dot products, the SVM solution can be expressed as follows:

$$f(\mathbf{x}; \boldsymbol{\vartheta}) = \sum_{l=1}^{L} \alpha_l y_l \, k(\mathbf{x}_l, \mathbf{x}) + b \ ,$$

where $k(\cdot,\cdot)$ is the dot product $\Phi(\cdot)\cdot\Phi(\cdot)$. More generally, $k(\cdot,\cdot)$ can be any kernel function that fulfills the Mercer conditions [Bur98], which ensure that, for any possible training set, the optimization problem is convex.

1.3.2 Kernels

A usual problem in machine learning is to extract features that are relevant for the classification task. For SVMs, choosing the features and choosing the kernel are equivalent problems, thanks to the so-called "kernel trick" mentioned above. The latter also permits to map $\mathbf{x}_l$ into potentially infinite dimensional feature spaces by avoiding the explicit computation of $\Phi(\mathbf{x}_l)$; it also reduces the computational load for mappings in finite but high dimension.

The two most well known kernels are the Radial Basis Function (RBF) kernel

$$k(\mathbf{x}_l,\mathbf{x}_{l'}) = \exp\left(\frac{-||\mathbf{x}_l-\mathbf{x}_{l'}||^2}{2\sigma^2}\right) \tag{1.8}$$

and the polynomial kernel

$$k(\mathbf{x}_l,\mathbf{x}_{l'}) = (a\,\mathbf{x}_l\cdot\mathbf{x}_{l'}+b)^p \ , \tag{1.9}$$

where σ, p, b, a are hyper-parameters that define the feature space.

Several SVM-based approaches have been proposed recently to tackle the speaker verification problem [WR03, CCR+06]. These approaches rely on constructing an ad-hoc kernel for the problem at hand. These kernels will be presented and evaluated after the following section that describes the details of the experimental methodology and the data that will be used to compare the various methods.

1.4 Benchmarking Methodology

In this section, we describe the methodology and the data used in all the experiments reported in this chapter. We first present the data splitting strategy that is used to imitate a realistic use of speaker verification systems. Then, we discuss the measures evaluating the performances of learning algorithms. Finally, we detail the database used to benchmark these algorithms, and the pre-processing that builds sequences of frames from waveform signals.

1.4.1 Data Splitting for Speaker Verification

A speaker verification problem is not a standard classification problem, since the objective is not to certify accesses from a pre-defined set of clients. Instead, we want to be able to authenticate new clients when they subscribe to the service, that is, we want to learn how to build new classifiers on the fly. Hence, a speaker verification system is evaluated by its ability to produce new classifiers with small test error. This is emulated by the data splitting process depicted in Figure 1.1.

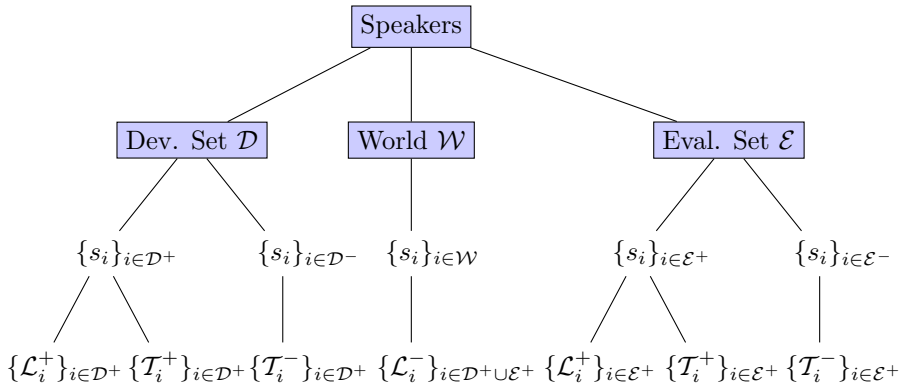

Figure 1.1: Split of the speaker population in three subsets, with the final decomposition in learning and test sets.

The root level gathers the population of speakers, which is split into three sub-populations, defined by their role in building classifiers: the development set $\mathcal{D}$, the world set $\mathcal{W}$ and the evaluation set $\mathcal{E}$. All accesses from the speakers of $\mathcal{W}$ will be used as the set of negative examples $\mathcal{L}_i^-$ for training the models responsible for authenticating client s_i, where s_i may belong either to the development set $\mathcal{D}$ or to the evaluation set $\mathcal{E}$. The sets $\mathcal{D}$ and $\mathcal{E}$ are further split into clients (resp. $\mathcal{D}^+$ and $\mathcal{E}^+$) and impostors (resp. $\mathcal{D}^-$ and $\mathcal{E}^-$) at the second level of the tree. The clients and the test impostors hence differ between the development and the evaluation sets.

The impostor accesses in $\mathcal{D}^-$ and $\mathcal{E}^-$ form the set of negative test examples $\mathcal{T}_i^-$, that is, "attempt data" from out-of-training impostors claiming identity s_i, where s_i belongs respectively to $\mathcal{D}^+$ and $\mathcal{E}^+$. Finally, at the third level of the tree, the accesses of client s_i are split to form the positive examples of the training set $\mathcal{L}_i^+$ (also known as the "enrollment data", usually a single access), and the set of positive "attempt data" $\mathcal{T}_i^+$ that play the role of out-of-training client accesses requiring authentication.

To summarize, the development set $\mathcal{D}$ is used jointly with $\mathcal{W}$ to train models and select their various hyper-parameters (such as the number of Gaussians, the MAP adaptation factor, kernel parameters, etc.). For each hyper-parameter, we define a range of possible values, and for each value, each client model is trained using the enrollment data $\mathcal{L}_i^+$ and the world data $\mathcal{L}_i^-$, before being evaluated with the positive and negative attempt data $\mathcal{T}_i^+$ and $\mathcal{T}_i^-$. We then select the value of the hyper-parameters that optimizes a given performance measure (the Equal Error Rate described below) on $\{\mathcal{T}_i^+ \cup \mathcal{T}_i^-\}$. Finally, the evaluation set $\mathcal{E}$ is used to train new client models using these hyper-parameters, and to measure the performance of the system on these new clients.

1.4.2 Performance Measures

The classification error rate is the most common performance measure in the machine learning literature, but it is not well suited to the type of problems encountered in speaker verification, where class priors are unknown and misclassification losses are unbalanced. Hence, a weighted version of the misclassification rate is used, where one distinguishes two kinds of errors: *False Rejection* (FR) which consists in rejecting a genuine client, and *False Acceptance* (FA) which consists in accepting an impostor. All the measures used in this chapter are based on the corresponding error rates: the *False Acceptance Rate* (FAR) is the number of FAs divided by the number of client accesses, and the *False Rejection Rate* (FRR) is the number of FRs divided by the number of impostor accesses.

As stated in the previous section, in practice, we aim at building a single system that is able to take decisions for all future users. The performance is measured globally, on the set of speakers of the evaluation set, by averaging the performance over all trials independently of the claimed identity.

In the speaker verification literature, a point often overlooked is that most of the results are reported with "a posteriori" measures, in the sense that the decision threshold Δ in Equation (1.1) is selected such that it optimizes some criterion on the evaluation set. We believe that this is unfortunate, and, in order to obtain unbiased results, we will use "a priori" measures, where the decision threshold Δ is selected on a development set, before seeing the evaluation set, and then applied to the evaluation data.

Common a posteriori measures include the Equal Error Rate (EER), where the threshold Δ is chosen such that (FAR=FRR), and the Detection Error Tradeoff (DET) curve [MDK$^+$97], which depicts FRR as a function of FAR when Δ varies. Note that the DET curve is is a non-linear transformation of the Receiver Operating Characteristic (ROC) curve [VT68]. The non-linearity is in fact a normal deviate, coming from the hypothesis that the scores of client accesses and impostor accesses follow a Gaussian distribution. These measures are perfectly legitimate for exploratory analysis or for tuning hyper-parameters on the development set and they are used in this purpose here. To avoid confusion with proper test results, we will only report DET curves computed on the development set. For test performance, we will use a priori measures: the Half Total Error Rate ($\text{HTER} = \frac{1}{2}(\text{FAR}(\Delta) + \text{FRR}(\Delta))$) and the Expected Performance Curve (EPC) [BMK05], which depicts the evaluation set HTER as a function of a trade-off parameter α. The latter defines a decision threshold, computed on the development set, by minimizing the following convex combination of development FAR and FRR:

$$\Delta^* = \arg\min_{\Delta} \Big(\alpha \cdot \text{FAR}(\Delta) + (1 - \alpha) \cdot \text{FRR}(\Delta) \Big) \ . \tag{1.10}$$

We will provide confidence intervals around HTER and EPC. In this chapter, we report confidence intervals computed at the 5% significance level, using an adapted version of the standard proportion test [BM04].

1.4.3 NIST Data

The NIST database is a subset of the database that was used for the *NIST 2005 and 2006 Speaker Recognition Evaluation*, which comes from the second release of the cellular switchboard corpus (Switchboard Cellular - Part 2) of the Linguistic Data Consortium. This data was used as development and evaluation sets while the training (negative) examples come from previous NIST campaigns. For both development and evaluation clients, there are about 2 minutes of telephone speech available to train the models and each test access was less than 1 minute long. Only male speakers were used. The development population consisted of 264 speakers, while the evaluation set contained 349 speakers. 219 different records were used as negative examples for the discriminant models. The total number of accesses in the development population is 13596 and 22131 for the evaluation set population with a proportion of 10% of true target accesses.

1.4.4 Pre-Processing

To extract input features, the original waveforms are sampled every 10ms with a window size of 20ms. Each sentence is parameterized using 24 triangular band-pass filters with a DCT transformation of order 16, complemented by their first derivative (delta) and the 10^{th} second derivative (delta-delta), the log-energy, the delta-log-energy and delta-delta-log-energy, for a total of 51 coefficients. The NIST database being telephone-based, the signal is band-pass filtered between 300 and 3400 Hz.

A simple silence detector, based on a two-components Gaussian mixture model, is used to remove all silence frames. The model is first learned on a random recording with land line microphone and adapted for each new sequence using the MAP adaptation algorithm. The sequences are then normalized in order to have zero mean and unit variance on each feature.

While the log-energy is important in order to remove the silence frames, it is known to be inappropriate to discriminate between clients and impostors. This feature is thus eliminated after silence removal, while its first derivative is kept. Hence, the speaker verification models are trained with 50 (51-1) features.

1.5 Kernels for Speaker Verification

One particularity of speaker verification is that patterns are sequences. An SVM based classification thus requires a kernel handling variable size sequences. Most solutions proposed in the literature use a procedure that converts the sequences into fixed size vectors that are processed by a linear SVM. Other sequence kernels allow embeddings in infinite-dimensional feature spaces [MB07]. However, compared to the mainstream approach, this type of kernels is computationally too demanding for long sequences. It will not be applied here, since the NIST database contains long sequences.

In the following we describe several approaches using sequence kernels. The most promising are then compared in Section 1.8.

1.5.1 Mean Operator Sequence Kernels

For kernel methods, a simple approach to tackle variable length sequences considers the following kernel between two sequences:

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \frac{1}{T_i T_j} \sum_{t=1}^{T_i} \sum_{u=1}^{T_j} k(\mathbf{x}_i^t, \mathbf{x}_j^u) \ , \tag{1.11}$$

where we denote by $K(\cdot,\cdot)$ a sequence kernel, $\bar{\mathbf{x}}_i$ is a sequence of size T_i and $\mathbf{x}_i^t$ is a frame of $\bar{\mathbf{x}}_i$. We thus apply a frame-based kernel $k(\cdot,\cdot)$ to all possible pairs of frames coming from the two input sequences $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}_j$.

As the kernel K represents the average similarity between all possible pairs of frames, it will be referred to as the mean operator sequence kernel. This kind of kernel has been applied successfully in other domains such as object recognition [BTF04]. Provided that $k(\cdot,\cdot)$ is positive-definite, the resulting kernel $K(\cdot,\cdot)$ is also positive-definite.

The sequences in the NIST database typically consist of several thousands of frames, hence the double summation in (1.11) is very costly. As the number of operations for each sequence kernel evaluation is proportional to the product of sequence lengths, such a computation typically requires an order of the million of operations. We thus will consider factorizable kernels $k(\cdot,\cdot)$, such that the mean operator sequence kernel (1.11) can be expressed as follows:

$$\begin{aligned} K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) &= \frac{1}{T_i T_j} \sum_{t=1}^{T_i} \sum_{u=1}^{T_j} \phi(\mathbf{x}_i^t) \cdot \phi(\mathbf{x}_j^u) \\ &= \left[\frac{1}{T_i} \sum_{t=1}^{T_i} \phi(\mathbf{x}_i^t) \right] \cdot \left[\frac{1}{T_j} \sum_{u=1}^{T_j} \phi(\mathbf{x}_j^u) \right] \ . \end{aligned} \tag{1.12}$$

When the dimension of the feature space is not too large, computing the dot product explicitly is not too demanding, and replacing the double summation by two single ones may result in a significant reduction of computing time.

Explicit polynomial expansions have been used in [Cam02, CCR+06, WR03]. In practice, the average feature vectors within brackets in (1.12) are used as input to a linear SVM. The GLDS (Generalized Linear Discriminant Sequence) kernel of Campbell departs slightly from a raw polynomial expansion, by using a normalization in the feature space:

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \frac{1}{T_i T_j} \Phi(\bar{\mathbf{x}}_i) \mathbf{\Gamma}^{-1} \Phi(\bar{\mathbf{x}}_j) \ , \tag{1.13}$$

where $\mathbf{\Gamma}$ defines a metric in the feature space. Typically, this is a diagonal approximation of the Mahalanobis metric, that is, $\mathbf{\Gamma}$ is a diagonal matrix whose

diagonal elements γ_k are the empirical variances[2] for each feature, computed over the training data.

The polynomial expansion sends d-dimensional frames to a feature space of dimension $(d+p)!/d!p! - 1$, where p is the degree of the polynomial. With our 50 input features, and for a polynomial of degree $p = 3$, the dimension of the feature space is 23426. For higher polynomial degrees and for other feature space of higher dimension, the computational advantage of the decomposition (1.12) disappears, and it is better to use explicit kernel in the form (1.11). We empirically show below that, for the usual representation of frames described in Section 1.4.4, the GLDS normalization in (1.13) is embedded in the standard polynomial kernel.

Let us define $k(\mathbf{x}_i, \mathbf{x}_j)$ as a polynomial kernel of the form $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p$, where p is the degree of the polynomial. After removing the constant term, the explicit expansion of this standard polynomial kernel involves $(d+p)!/d!p! - 1$ terms that can be indexed by $\mathbf{r} = (r_1, r_2, ..., r_d)$, such that

$$\phi_{\mathbf{r}}(\mathbf{x}) = \sqrt{c_{\mathbf{r}}} x_1^{r_1} x_2^{r_2} ... x_d^{r_d} \ ,$$

$$\text{where} \quad \sum_{i=1}^{d} r_i = p \ , \quad r_i \geq 0 \ , \quad \text{and} \quad c_{\mathbf{r}} = \frac{p!}{r_1! r_2! ... r_{d+1}!} \ .$$

In the above equations, $\sqrt{c_{\mathbf{r}}}$ has exactly the same role as the $1/\sqrt{\gamma_k}$ coefficients on the diagonal of $\mathbf{\Gamma}^{-1/2}$ in Equation (1.13). In Figure 1.2, we compare these coefficient values, where the normalization factors $1/\sqrt{\gamma_k}$ are estimated on two real datasets, after a polynomial expansion of degree 3. The values are very similar, with highs and lows on the same monomial. In fact, the performance of the two approaches obtained on the development set of NIST are about the same, as shown by the DET curves given in Figure 1.3.

Even if this approach is simple and easy to use, the accuracy can be improved by introducing priors. In fact, to train a client model very few positive examples are available. Thus, if we can put pieces of information collected on large set of speakers into the SVM model, as done for the GMM system, we can expect an improvement. One can for example try to include the world model in the kernel function as proposed in the next Section.

1.5.2 Fisher Kernels

Jaakkola and Haussler proposed a principled means for building kernel functions from generative models: the Fisher kernel [JH98]. In this framework, which has been applied to speaker verification by [WR05a], the generative model is used to specify the similarity between pairs of examples, instead of the usual practice where it is used to provide a likelihood score, which measures how well the example fits the model. Put it another way, a Fisher kernel utilizes a generative model to measure the differences in the generative process between pairs of examples instead of the differences in posterior probabilities.

[2]The constant feature is removed from the feature space prior to normalization.

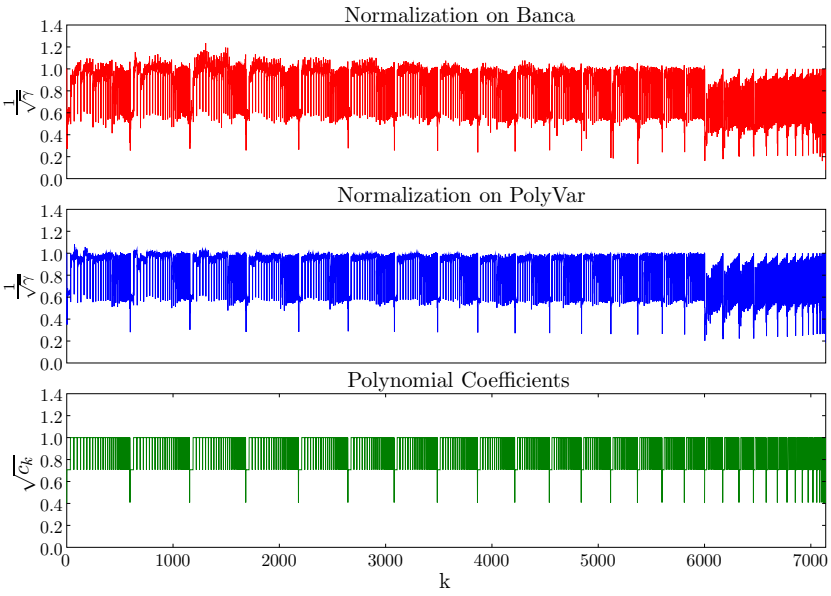

Figure 1.2: Coefficient values of polynomial terms, as computed on two different datasets (Banca and PolyVar), compared to the c_k polynomial coefficients.

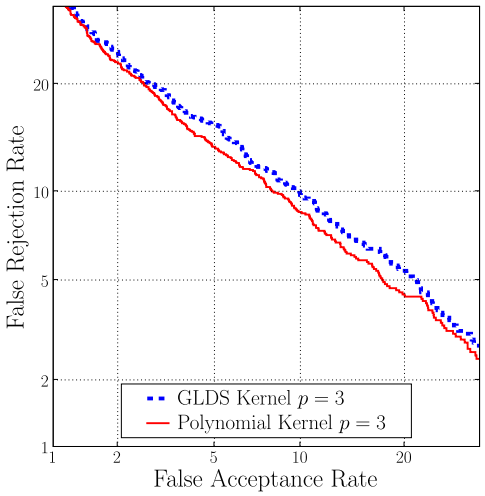

Figure 1.3: DET curves on the development set of the NIST database comparing the explicit polynomial expansion (noted as "GLDS kernel $p = 3$ in the legend), and the principled polynomial kernel (noted "Polynomial kernel $p = 3$").

The key ingredient of the Fisher kernel is the vector of Fisher scores:

$$\mathbf{u}_{\bar{\mathbf{x}}} = \nabla_{\boldsymbol{\theta}} \log p(\bar{\mathbf{x}}|\boldsymbol{\theta}) \ ,$$

where $\boldsymbol{\theta}$ denotes here the parameters of the generative model, and $\nabla_{\boldsymbol{\theta}}$ is the gradient with respect to $\boldsymbol{\theta}$. The Fisher scores quantify how much each parameter contributes to the generation of example $\bar{\mathbf{x}}$.

The Fisher kernel itself is given by:

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \mathbf{u}_{\bar{\mathbf{x}}_i}^T \mathbf{I}(\boldsymbol{\theta})^{-1} \mathbf{u}_{\bar{\mathbf{x}}_j} \ , \tag{1.14}$$

where $\mathbf{I}(\boldsymbol{\theta})$ is the Fisher information matrix at $\boldsymbol{\theta}$, that is, the covariance matrix of Fisher scores:

$$\mathbf{I}(\boldsymbol{\theta}) = \mathbb{E}_{\bar{\mathbf{x}}}(\mathbf{u}_{\bar{\mathbf{x}}} \mathbf{u}_{\bar{\mathbf{x}}}^T) \ , \tag{1.15}$$

where we used that $\mathbb{E}_{\bar{\mathbf{x}}}(\mathbf{u}_{\bar{\mathbf{x}}}) = 0$. The Fisher kernel (1.14) can thus be interpreted as a Mahalanobis distance between two Fisher scores.

Another interpretation of the Fisher kernel is based on the representation of a parametric class of generative models as a Riemannian manifold [JH98]. Here, the vector of Fisher scores defines a tangent direction at a given location, that is, at a given model parameterized by $\boldsymbol{\theta}$. The Fisher information matrix is the local metric at this given point, which defines the distance between the current model $p(\bar{\mathbf{x}}|\boldsymbol{\theta})$ and its neighbors $p(\bar{\mathbf{x}}|\boldsymbol{\theta} + \boldsymbol{\delta})$. The (squared) distance $d(\boldsymbol{\theta}, \boldsymbol{\theta} + \boldsymbol{\delta}) = \frac{1}{2}\boldsymbol{\delta}^T \mathbf{I} \boldsymbol{\delta}$ approximates the Kullback-Leibler divergence between the two models. Note that, unlike the Kullback-Leibler divergence, the Fisher kernel (1.14) is symmetric. It is also positive-definite since the Fisher information matrix $\mathbf{I}(\boldsymbol{\theta})$ is obviously positive-definite at $\boldsymbol{\theta}$.

Fisher Kernels for GMMs

In the MAP framework, the family of generative models we consider is the set of Gaussian mixtures (1.4) that differ in their mean vectors $\boldsymbol{\mu}_m$. Hence, a relevant dissimilarity between examples will be measured by the Fisher scores computed on these vectors $\mathbf{u}_{\bar{\mathbf{x}}} = (\nabla_{\boldsymbol{\mu}_1}^T \log p(\bar{\mathbf{x}}|\boldsymbol{\theta}), \ldots, \nabla_{\boldsymbol{\mu}_M}^T \log p(\bar{\mathbf{x}}|\boldsymbol{\theta}))^T$, where

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}_m} \log p(\bar{\mathbf{x}}|\boldsymbol{\theta}) &= \sum_{t=1}^{T} \nabla_{\boldsymbol{\mu}_m} \log \sum_{m'=1}^{M} \pi_{m'} \mathcal{N}(\mathbf{x}^t|\boldsymbol{\mu}_{m'}, \boldsymbol{\sigma}_{m'}) \\
&= \sum_{t=1}^{T} P(m|\mathbf{x}^t) \nabla_{\boldsymbol{\mu}_m} \left(-\frac{1}{2}(\mathbf{x}^t - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-2} (\mathbf{x}^t - \boldsymbol{\mu}_m) \right) \\
&= \sum_{t=1}^{T} P(m|\mathbf{x}^t) \, \boldsymbol{\Sigma}_m^{-2} (\mathbf{x}^t - \boldsymbol{\mu}_m) \ .
\end{aligned} \tag{1.16}$$

Using definition (1.15), the Fisher information matrix can be expressed block-wise, with $M \times M$ blocks of size $d \times d$:

$$\mathbf{I} = (\mathbf{I}_{m,m'})_{1 \leq m \leq M, 1 \leq m' \leq M} \ ,$$

with

$$\mathbf{I}_{m,m'} = \mathbb{E}_{\bar{\mathbf{x}}}\Big[\sum_{t=1}^{T}\sum_{u=1}^{T} P(m|\mathbf{x}^t)P(m'|\mathbf{x}^u)\boldsymbol{\Sigma}_m^{-2}(\mathbf{x}^t-\boldsymbol{\mu}_m)(\mathbf{x}^u-\boldsymbol{\mu}_{m'})^T\boldsymbol{\Sigma}_{m'}^{-2}\Big] \ . \quad (1.17)$$

There is no simple analytical expression of this expectation, due, among other things, to the product $P(m|\mathbf{x}^t)P(m'|\mathbf{x}^u)$. Hence, several options are possible:

1. ignore the information matrix in the computation of the Fisher kernel (1.14). This option, mentioned by Jaakkola and Haussler as a simpler suitable substitute, is often used in the application of Fisher kernels;

2. approximate the expectation in the definition of Fisher information by Monte Carlo sampling.

3. approximate the product $P(m|\mathbf{x}^t)P(m'|\mathbf{x}^u)$ by a simpler expression in (1.17). For example, if we assume that the considered GMM performs hard assignments of frames to mixture components, then $P(m|\mathbf{x}^t)P(m'|\mathbf{x}^u)$ is null if $m \neq m'$. Furthermore, this product is also null for $m = m'$ when $\mathbf{x}^t$ or $\mathbf{x}^u$ is generated from another component of the mixture distribution, otherwise, we have $P(m|\mathbf{x}^t)P(m'|\mathbf{x}^u) = 1$. Let g_m denote the function such that $g_m(\mathbf{x},\mathbf{y}) = \boldsymbol{\Sigma}_m^{-2}(\mathbf{x}-\boldsymbol{\mu}_m)(\mathbf{y}-\boldsymbol{\mu}_{m'})^T\boldsymbol{\Sigma}_{m'}^{-2}$ if $P(m|\mathbf{x}) = P(m|\mathbf{y}) = 1$ and $g_m(\mathbf{x},\mathbf{y}) = 0$ otherwise. With this notation and the above approximations, (1.17) reads

$$\begin{aligned}
\mathbf{I}_{m,m'} &\simeq 0 \text{ if } m \neq m' \\
\mathbf{I}_{m,m} &\simeq \mathbb{E}_{\bar{\mathbf{x}}}\Big[\sum_{t=1}^{T}\sum_{u=1}^{T} g_m(\mathbf{x}^t,\mathbf{x}^u)\Big] \\
&\simeq \mathbb{E}_{\mathbf{x}}\left[g_m(\mathbf{x},\mathbf{x})\right]\mathbb{E}_T\left[T\right] \\
&\simeq \boldsymbol{\Sigma}_m^{-2}\mathbb{E}_T\left[T\right] \ .
\end{aligned}$$

The unknown constant $\mathbb{E}_T[T]$ is not relevant and can been dropped from the implementation of this approximation to the Fisher kernel.

We now introduce some definitions with the following scenario. Suppose that we trained the GMM world model on a large set of speakers, resulting in parameters $\boldsymbol{\theta}_0 = \{\boldsymbol{\mu}_m^0, \boldsymbol{\sigma}_m, \pi_m\}_{m=1}^M$. We then use this GMM as an initial guess for the model for client s_i. If, as in the MAP framework, the client model differs from the world model in the mean vector only, then, after one EM update, the training sequence $\bar{\mathbf{x}}_i$ will result in the following estimates

$$\begin{aligned}
\boldsymbol{\mu}_m^i &= \frac{1}{n_{i,m}}\sum_{t=1}^{T_i}\mathbf{x}_i^t P(m|\mathbf{x}_i^t) \ , \\
\text{where } n_{i,m} &= \sum_{t=1}^{T_i} P(m|\mathbf{x}_i^t) \ .
\end{aligned}$$

Hence, $n_{i,m}$ is the effective number of frames used to compute $\boldsymbol{\mu}_m^i$, that is, the sum of the membership of all frames of $\bar{\mathbf{x}}_i$ to component m. These definitions of $\boldsymbol{\mu}_m^i$ and $n_{i,m}$ are convenient for expressing Fisher scores, when the reference generative model is the world model parameterized by $\boldsymbol{\theta}_0$

$$\begin{aligned} \nabla_{\boldsymbol{\mu}_m} \log p(\bar{\mathbf{x}}_i|\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} &= \boldsymbol{\Sigma}_m^{-2} \sum_{t=1}^{T_i} P(m|\mathbf{x}_i^t)\,(\mathbf{x}_i^t - \boldsymbol{\mu}_m) \\ &= n_{i,m}\boldsymbol{\Sigma}_m^{-2}(\boldsymbol{\mu}_m^i - \boldsymbol{\mu}_m^0)\ . \end{aligned} \tag{1.18}$$

With the approximations of the Fisher information discussed above, the kernel is expressed as:

1. for the option where the Fisher information matrix is ignored:

$$\begin{aligned} K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) &= \mathbf{u}_{\bar{\mathbf{x}}_i}^T \mathbf{u}_{\bar{\mathbf{x}}_j} \\ &= \sum_{m=1}^{M} \left(n_{i,m}\boldsymbol{\Sigma}_m^{-2}(\boldsymbol{\mu}_m^i - \boldsymbol{\mu}_m^0)\right)^T \left(n_{j,m}\boldsymbol{\Sigma}_m^{-2}(\boldsymbol{\mu}_m^j - \boldsymbol{\mu}_m^0)\right) \end{aligned}$$

2. for the option where the Fisher information matrix is approximated by Monte Carlo integration: here, for computational reasons, we only consider a block-diagonal approximation $\widehat{\mathbf{I}}$, where

$$\widehat{\mathbf{I}} = (\widehat{\mathbf{I}}_{m,m'})_{1\le m\le M, 1\le m'\le M}\ ,$$

with

$$\begin{aligned} \widehat{\mathbf{I}}_{m,m'} &= 0 \text{ if } m \neq m' \\ \widehat{\mathbf{I}}_{m,m} &= \frac{1}{n}\sum_t P(m|\mathbf{x}^t)^2 \boldsymbol{\Sigma}_m^{-2}(\mathbf{x}^t - \boldsymbol{\mu}_m^0)(\mathbf{x}^t - \boldsymbol{\mu}_m^0)^T \boldsymbol{\Sigma}_m^{-2}\ , \end{aligned}$$

where n is the number of random draws of $\mathbf{x}^t$ generated from the world model.

We then have:

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \sum_{m=1}^{M} \left(n_{i,m}\boldsymbol{\Sigma}_m^{-2}(\boldsymbol{\mu}_m^i - \boldsymbol{\mu}_m^0)\right)^T \widehat{\mathbf{I}}_{m,m}^{-1} \left(n_{j,m}\boldsymbol{\Sigma}_m^{-2}(\boldsymbol{\mu}_m^j - \boldsymbol{\mu}_m^0)\right)$$

3. for the option where the Fisher information matrix is approximated analytically

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \sum_{m=1}^{M} \left(n_{i,m}\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{\mu}_m^i - \boldsymbol{\mu}_m^0)\right)^T \left(n_{j,m}\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{\mu}_m^j - \boldsymbol{\mu}_m^0)\right)$$

These three variants of the Fisher kernel are compared in Figure 1.4, which compares the DET curves obtained on the development set of the NIST database. The three curves almost overlap, confirming that ignoring the information matrix in the Fisher kernel is not harmful in our setup.

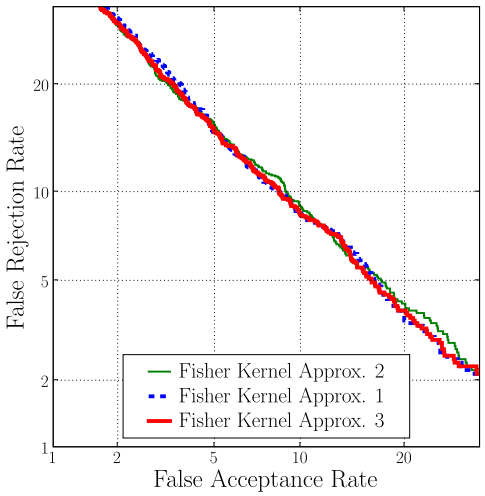

Figure 1.4: DET curves on the development set of the NIST database comparing the three different approximations of the Fisher information matrix.

1.5.3 Beyond Fisher Kernels

The previous experimental results confirm that the main ingredient of the Fisher kernel is the Fisher score. The latter is based on a probabilistic model viewed through the log-likelihood function. We can depart from the original setup described above, by using other models and/or score. Some alternative approaches has been already investigated, for example [WR05b], uses scores based on a log likelihood ratio between the world model and the adapted client model. We describe below a very simple modification of the scoring function that brings noticeable improvements in performances.

Normalized Fisher Scores

We saw in Section 1.2.2 that the scores used for classifying examples are normalized, in order to counterbalance the exponential decrease of likelihoods with sequence lengths. Using the normalized likelihood leads to the following Fisher-like kernel

$$\begin{aligned} K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) &= \frac{1}{T_i T_j} \mathbf{u}_{\bar{\mathbf{x}}_i}^T \mathbf{u}_{\bar{\mathbf{x}}_j} \\ &= \sum_{m=1}^{M} \left(\frac{n_{i,m}}{T_i} \boldsymbol{\Sigma}_m^{-2} (\boldsymbol{\mu}_m^i - \boldsymbol{\mu}_m^0) \right)^T \left(\frac{n_{j,m}}{T_j} \boldsymbol{\Sigma}_m^{-2} (\boldsymbol{\mu}_m^j - \boldsymbol{\mu}_m^0) \right) \end{aligned}$$

Here also, one may consider several options for approximating the Fisher information matrix, but the results displayed in Figure 1.4 suggest it is not worth

Table 1.1: EERs (the lower the better) on the development set of the NIST database, comparing Fisher kernel (approximation 3), the normalized Fisher kernel.

	Fisher	Normalized Fisher
EER (%)	9.3	8.2
95% confidence	±0.9	±0.8
# Support Vectors	37	32

pursuing this road further. Table 1.1 and Figure 1.5 compare empirically the

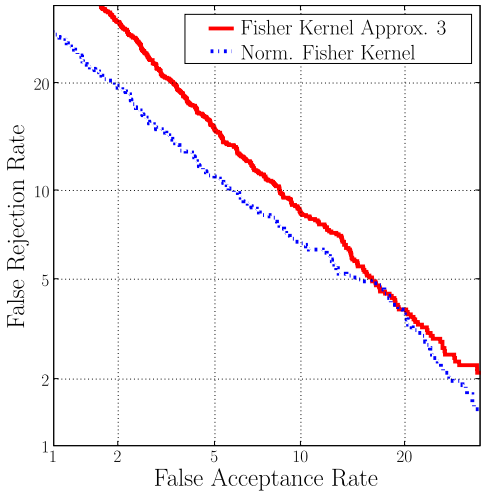

Figure 1.5: DET curves on the development set of the NIST database for Fisher kernel (approximation 3) and normalized Fisher kernel.

Fisher kernel (approximation 3) with the normalized Fisher kernel. Including a normalization seems have a positive impact on the accuracy. Thus other kind of scores should be explored.

GMM Supervector Linear Kernel

The Fisher kernel is a similarity based on the differences in the generation of examples. In this matter, it is related to the GMM Supervector Linear Kernel (GSLK) proposed by Campbell *et al.* [CSR06].

The GSLK approximates the Kullback-Leibler divergence that measures the

Table 1.2: EERs (the lower the better) on the development set of the NIST database, comparing GSLK and the normalized Fisher kernel.

	GSLK	Normalized Fisher
EER (%)	7.9	8.2
95% confidence	±0.8	±0.8
# Support Vectors	34	32

dissimilarity between two GMMs, each of one being obtained by adapting the world model to one example of the pair $(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$. Hence, instead of looking at how a single generative process differs for each example of the $(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ pair, GSLK looks at the difference between pairs of generative models. The GSLK is given by:

$$K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \sum_{m=1}^{M} \left(\sqrt{\pi_m}\boldsymbol{\Sigma}_m^{-1}\left(\tau_{i,m}\boldsymbol{\mu}_m^i + (1-\tau_{i,m})\boldsymbol{\mu}_m^0\right)\right)^T \cdot \left(\sqrt{\pi_m}\boldsymbol{\Sigma}_m^{-1}\left(\tau_{j,m}\boldsymbol{\mu}_m^j + (1-\tau_{j,m})\boldsymbol{\mu}_m^0\right)\right) \ ,$$

where $\tau_{i,m}$ is the adaptation factor for the mixture component m adapted with sequence $\bar{\mathbf{x}}_i$, as defined in (1.5). The MAP relevant factor r is chosen by cross-validation, as in GMM based text-independent speaker verification systems [RQD00].

The Fisher kernel and GSLK are somewhat similar scalar products, with the most noticeable difference being that the Fisher similarity is based on difference from the reference $\boldsymbol{\mu}^0$ whereas the GSLK kernel above is based on a convex combination of the observations and the reference $\boldsymbol{\mu}^0$ that has no obvious interpretation. Both are an approximation of the KL divergence as mentioned in Section 1.5.2. The difference is that GSLK compare two adapted distributions when the Fisher kernel compare the world model to the updated model using the access data.

Table 1.2 and Figure 1.6 compare empirically GSLK with the normalized Fisher kernel. There is no significant difference between GSLK and the normalized Fisher kernel.

1.6 Parameter Sharing

The text-independent speaker verification problem is actually a set of several binary classification problems, one for each client of the system. Although few positive examples are available for each client, the overall number of available positive examples may be large. Hence, techniques that share information between classification problems should be beneficial. We already mentioned such

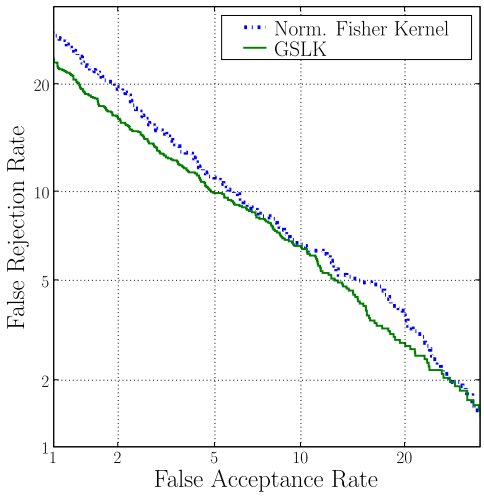

Figure 1.6: DET curves on the development set of the NIST database for GSLK and normalized Fisher kernel.

a technique: the MAP adaptation scheme that trains a single world model on a common data set, and uses it as a prior distribution over the parameters to train a GMM for each client. Here, the role of the world model is to bias each client model towards a reference speaker model. This bias amounts to a soft sharing of parameters.

Additional parameter sharing techniques are now used in discriminant approaches. In the following, we discuss one of them, the *Nuisance Attribute Projection* (NAP).

1.6.1 Nuisance Attribute Projection

The *Nuisance Attribute Projection* (NAP) approach [SQC04] looks for a linear subspace such that similar accesses (that is, accesses coming from the same client or from the same channel, etc) are near each others. In order to refrain from finding an obvious bad solution, the dimension of the target subspace is controlled by cross-validation. This transformation is learned on a large set of clients (similarly to learning a generic GMM in the generative approach). After this step is performed, a standard linear SVM is usually trained for each new client over the transformed access data. This approach provided very good performance in recent NIST evaluations.

More specifically, assume each access sequence $\bar{\mathbf{x}}$ is mapped into a fixed-size feature space through some transformation $\Phi(\bar{\mathbf{x}})$ such as the one used in the GLDS kernel. Let $\mathbf{W}^c$ be a proximity matrix encoding, for each pair of accesses $(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$, that these sequences were recorded over the same channel ($W^c_{i,j} = 0$) or

not ($W^c_{i,j} = 1$). The NAP approach then consists in finding a projection matrix $\mathbf{P}^\star$ such that

$$\mathbf{P}^\star = \arg\min_{\mathbf{P}} \sum_{i,j} W^c_{i,j} \|\mathbf{P}(\Phi(\bar{\mathbf{x}}_i) - \Phi(\bar{\mathbf{x}}_j))\|^2 \tag{1.19}$$

among orthonormal projection matrices of a given rank. Hence $\mathbf{P}^\star$ minimizes the average difference between accesses from differing channels, in the feature space. Similarly, a second matrix $\mathbf{W}^s$ could encode the fact that two accesses come from the same speaker. A combination between these prior knowledge could be encoded as follows

$$\mathbf{W} = \alpha \mathbf{W}^c - \gamma \mathbf{W}^s \ , \tag{1.20}$$

with α and γ hyper-parameters to tune, and $\mathbf{P}^\star$ found to minimize equation (1.19) with $\mathbf{W}$ instead of $\mathbf{W}^c$.

As stated earlier, $\mathbf{P}^\star$ is then used to project each access $\Phi(\bar{\mathbf{x}})$ into a feature subspace where, for each client, a linear SVM is used to discriminate client and impostor accesses. As shown in Table 1.3 and Figure 1.7, NAP brings significant improvement when combined with the GSLK kernel. On the other hand, the number of support vectors grows also significantly. This can be interpreted that now all accesses are in the same space and are independent to the channel and thus more training impostors are good candidates.

Table 1.3: EERs (the lower the better) on the development set of the NIST database, comparing an SVM classifier with GSLK with and without NAP (polynomial kernel of degree 3).

	GSKL	GSLK with NAP
EER (%)	7.9	5.8
95% confidence	±0.8	±0.6
# Support Vectors	34	59

Although the approach has shown to yield very good performance results, we believe that there is still room for improvements, since $\mathbf{P}^\star$ is not selected using the criterion that is directly related to the task. Minimizing the average squared distance between accesses of the same client (or accesses of different channel) is likely to help classification, but it would also be relevant to do something about accesses from different clients, such as moving them away for instance.

1.6.2 Other Approaches

Another recent approach that also goes in the same direction and that obtains state-of-the-art performance similar to the NAP approach is the Bayesian Factor Analysis approach [KBD05]. In this case, one assumes that the mean vector

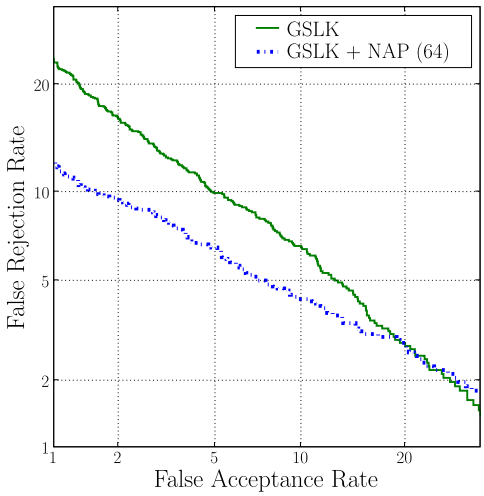

Figure 1.7: DET curves on the development set of the NIST database with GSLK with and without NAP.

of a client model is a linear combination of a generic mean vector, the mean vector of the available training data for that client, and the mean vector of the particular channel used in this training data. Once again, the linear combination parameters are trained on a large amount of access data, involving a large amount of clients. While this approach is nicely presented theoretically (and obtains very good empirical performance), it still does not try to find the optimal parameters of client models and linear combination by taking into account the global cost function.

Another very promising line of research that has emerged in machine learning relates to the general problem of learning a similarity metric [CHL05, WBS05, Leb06]. In this setting, where the learning algorithm relies on the comparison of two examples, one can set aside some training examples to actually learn what would be a good metric to compare pairs of examples. Obviously, in the SVM world, this relates to learning the kernel itself [CKS02, LCB+04].

In the context of discriminant approaches to speaker verification, none of these techniques have been tried, to the best of our knowledge. Using a large base of accesses for which one knows the correct identity, one could for instance train a parametric similarity measure that would assess whether two accesses are coming from the same person or not. That could be done efficiently by stochastic gradient descent using a scheme similar to the so-called *Siamese neural network* [CHL05] and a margin criterion with proximity constraints.

1.7 Is the Margin Useful for this Problem?

The scarcity of positive training examples in speaker verification explains the great improvements that pertain to parameter sharing techniques. In this section, we question whether this specificity also hinders large margin methods to improve upon more simple approaches.

The K-Nearest Neighbors (KNN) algorithm [DH73] is probably the simplest and the most known non-parametric classifier. Instead of learning a decision boundary, decisions are computed on-the-fly for each test access, by using the k nearest labelled sequences in the database as "experts", whose votes are aggregated to make up the decision on the current access.

In the weighted KNN [Dud86] variant, the votes of the nearest neighbors are weighted according to their distance to the query:

$$f(\bar{\mathbf{x}}_j) = \sum_{i=1}^{k} y_i w_i \ \text{, with } w_i = \begin{cases} 1 & \text{if } d(j,k) = d(j,1) \\ \frac{d(j,k)-d(j,i)}{d(j,k)-d(j,1)} & \text{otherwise,} \end{cases} \tag{1.21}$$

where the sum runs over the k neighbors of the query $\bar{\mathbf{x}}_j$, $y_i \in \{-1, 1\}$ determines whether the neighbor's access is from a client ($y_i = 1$) or an impostor ($y_1 = -1$), and $d(j, i)$ is the distance from $\bar{\mathbf{x}}_j$ to its i^{th} neighbor.

One can then use kernels to define distances, as follows:

$$d(i,j) = \sqrt{K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_i) - 2K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) + K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_j)} \ , \tag{1.22}$$

but it is often better to normalize the data also in the feature space so that they have unit norm, as follows,

$$K_{norm}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = \frac{K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)}{\sqrt{K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_i)\ K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_j)}} \ , \tag{1.23}$$

which leads to the final distance measure used in the experiments:

$$d_{norm}(i,j) = \sqrt{2 - 2\frac{K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)}{\sqrt{K(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_i)\ K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_j)}}} \ . \tag{1.24}$$

Table 1.4 and Figure 1.8 compares the normalized Fisher score with NAP approach followed by either an SVM or a KNN, and as can be seen, the KNN approach yields similar if not better performance than the SVM approach. Furthermore, the KNN has several advantages with respect the SVMs: there is no training session, KNN can easily approximate posterior probabilities and do not rely on potentially constraining Mercer conditions to work. On the other hand, the test session might be longer as finding nearest neighbors needs to be efficiently implemented.

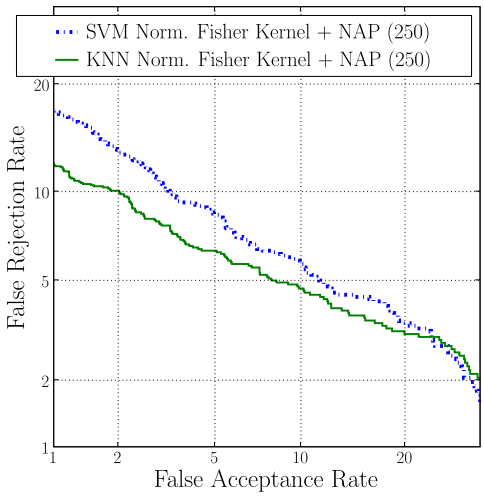

Figure 1.8: DET curves on the development set of the NIST database comparing Fisher normalized kernel with NAP for KNN and SVM.

Table 1.4: EERs (the lower the better) on the development set of the NIST database, comparing the Fisher normalized kernel with NAP (250) for KNN and SVM.

	SVM	KNN
EER (%)	6.7	5.3
95% confidence	±0.7	±0.7
# Support Vectors	47	-

1.8 Comparing All Methods

As a final experiment, we have compared all the proposed approaches and now report the results on the evaluation set. Figure 1.9 compares a state-of-the-art diagonal GMM with an SVM using a GSLK kernel with NAP, and also with a KNN based on the normalized Fisher kernel with NAP.

In this experiment, the following set of hyper-parameters were tuned according to the EER obtained on the development set:

- the number of neighbors K in the KNN approach, was varied between 20 and 200, with optimal value: 100;
- the size of $\mathbf{P}$, the transformed space in NAP for the GSLK kernel, was

varied between 40 and 250, with optimal value: 64;

- the size of $\mathbf{P}$, the transformed space in NAP for the Fisher kernel, was varied between 40 and 400, with optimal value: 250;

- the number of Gaussians in the GMM used for the GSLK and Fisher kernel approaches was varied between 100 and 500, with optimal value: 200;

- all other parameters of the state-of-the-art diagonal GMM baseline were taken from previously published experiments.

The GMM yields the worst performance, probably partly because no channel compensation method is used (while the others use NAP). KNN and SVM performances do not differ significantly, hence the margin does not appear to be at all necessary for speaker verification.

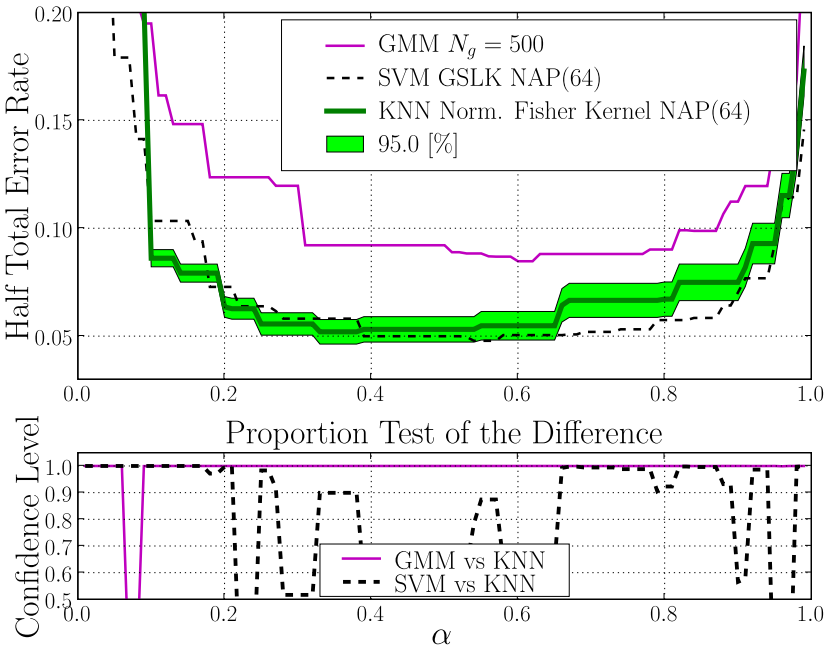

Figure 1.9: Expected Performance Curve (the lower, the better) on the evaluation set of the NIST database comparing GMM with T-norm, SVM with a GSLK kernel and NAP, SVM Fisher normalized kernel with NAP and KNN with a Fisher normalized kernel with NAP.

Table 1.5: Final results on the evaluation set of the NIST database

	GMM	SVM GSKL NAP	KNN Normalized Fisher NAP
HTER (%)	10.2	5.4	5.5
95% Conf.	±0.7	±0.5	±0.5

1.9 Conclusion

In this chapter, we have presented the task of text independent speaker verification. We have shown that the traditional method to approach this task is through a generative approach based on Gaussian Mixture Models.

We have then presented a discriminative framework for this task, and presented several recent approaches in this framework, mainly based on Support Vector Machines. We have presented various kernels adapted to the task, including the GLDS, GSLK and Fisher kernels. While many of the proposed kernels in the literature were proposed in some heuristic way, including the GLDS and GSLK kernels, we have shown the relation between the principled polynomial kernel and the GLDS kernel, as well as the relation between the principled Fisher kernel and the GSLK kernel. We have then shown that in order for SVMs to perform at a state-of-the-art level, parameter sharing in one way or another was necessary. Approaches such as NAP or Bayesian Factor Analysis were designed for that purpose and indeed helped SVMs to reach better performance.

Finally, we have questioned the main purpose of using SVMs, which maximize the margin in the feature space. We have tried instead a plain KNN approach, which yielded similar performance. This simple experiment shows that future research should concentrate more on better modelling of the distance measure, rather than on maximizing the margin.

A drawback of the current approaches is that they are made of various blocks (feature extraction, feature normalization, distance measure, etc) which were all trained using a separate ad-hoc criterion. Ultimately, a system that would train all these steps in a single framework to optimize the final objective should perform better, but more research is necessary to reach that goal.

In order to foster more research in this domain, an open-source version of the C++ source code used to performed all experiments described in this chapter have been made available at `http://speaker.abracadoudou.com`.

Acknoledgements

This work was supported in part by the Swiss National Science Foundation through the Swiss National Center of Competence in Research (NCCR) on Interactive Multi-modal Information Management (IM2), through the A-Vision

CTI project CTI:#8876.1-PFES-ES and by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886.

Bibliography

[BM04] S. Bengio and J. Mariéthoz. A statistical significance test for person authentication. In *Proceedings of Odyssey 2004: The Speaker and Language Recognition Workshop*, pages 237–240, 2004.

[BMK05] S. Bengio, J. Mariéthoz, and M. Keller. The expected performance curve. In *International Conference on Machine Learning, ICML, Workshop on ROC Analysis in Machine Learning*, 2005.

[BTF04] S. Boughorbel, J. P. Tarel, and F. Fleuret. Non-Mercer kernel for SVM object recognition. In *British Machine Vision Conference*, 2004.

[Bur98] C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.

[Cam02] W.M. Campbell. Generalized linear discriminant sequence kernels for speaker recognition. In *Proc IEEE International Conference on Audio Speech and Signal Processing*, pages 161–164, 2002.

[CCR+06] W.M. Campbell, J.P. Campbell, D.A. Reynolds, E. Singer, and P.A. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20(2-3):125–127, 2006.

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[CKS02] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems, NIPS*, 2002.

[CSR06] W.M. Campbell, D.E. Sturim, and D.A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *Signal Processing Letters, IEEE*, 13(5):308–311, March 2006.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[Dud86] S. A. Dudani. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(4):325–327, 1986.

[GL94] J. L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observation of markov chains. In *IEEE Transactions on Speech Audio Processing*, volume 2, pages 291–298, April 1994.

[JH98] T.S Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing*, 11:487–493, 1998.

[Joa02] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers, Dordrecht, NL, 2002.

[KBD05] P. Kenny, G. Boulianne, and P. Dumouchel. Eigenvoice modeling with sparse training data. *IEEE Transactions on Speech and Audio Processing*, 13(3), 2005.

[LCB+04] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research, JMLR*, 5:27–72, 2004.

[Leb06] G. Lebanon. Metric learning for text documents. *IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI*, 28:497–508, 2006.

[MB07] Johnny Mariéthoz and Samy Bengio. A kernel trick for sequences applied to text-independent speaker verification systems. *Pattern Recognition*, 2007. IDIAP-RR 05-77.

[MDK+97] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proceedings of Eurospeech'97, Rhodes, Greece*, pages 1895–1898, 1997.

[MKLB98] H. Melin, J.W. Koolwaaij, J. Lindberg, and F. Bimbot. A comparative evaluation of variance flooring techniques in hmm-based speaker verification. In *ICSLP 1998*, pages 1903–1906, 1998.

[PV98] M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Transaction PAMI*, 20:637–646, 1998.

[RQD00] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1–3), 2000.

[SQC04] A. Solomonoff, C. Quillen, and W.M. Campbell. Channel compensation for SVM speaker recognition. In *Proceedings of Odyssey 2004: The Speaker and Language Recognition Workshop*, pages 57–62, 2004.

[Vap00] V. N. Vapnik. *The nature of statistical learning theory.* Springer, second edition, 2000.

[VT68] H. L. Van Trees. *Detection, Estimation and Modulation Theory, vol. 1.* Wiley, New York, 1968.

[WBS05] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems, NIPS*, 2005.

[WR03] V. Wan and S. Renals. Support vector machine speaker verification methodology. In *IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP*, pages 221–224, 2003.

[WR05a] Vincent Wan and Steve Renals. Speaker verification using sequence discriminant support vector machines. *IEEE Transactions on Speech and Audio Processing*, 13(2):203–210, 2005.

[WR05b] Vincent Wan and Steve Renals. Speaker verification using sequence discriminant support vector machines. *IEEE Transactions on Speech and Audio Processing*, 12(2):203–210, March 2005.