The Speaker and Language Recognition Workshop (Odyssey 2018)
26-29 June 2018, Les Sables d'Olonne, France

Gaussian meta-embeddings for efficient scoring of a heavy-tailed PLDA model

Niko Brümmer[1], *Anna Silnova*[2], *Lukáš Burget*[2] *and Themos Stafylakis*[3]

1. Nuance Communications, South Africa
2. Brno University of Technology, Czech Republic
3. Computer Vision Lab, University of Nottingham & Omilia Conversational Intelligence, UK

niko.brummer@gmail.com, isilnova@fit.vutbr.cz

Abstract

Embeddings in machine learning are low-dimensional representations of complex input patterns, with the property that simple geometric operations like Euclidean distances and dot products can be used for classification and comparison tasks. We introduce meta-embeddings, which live in more general inner product spaces and which are designed to better propagate uncertainty through the embedding bottleneck. Traditional embeddings are trained to maximize between-class and minimize within-class distances. Meta-embeddings are trained to maximize relevant information throughput. As a proof of concept in speaker recognition, we derive an extractor from the familiar generative Gaussian PLDA model (GPLDA). We show that GPLDA likelihood ratio scores are given by Hilbert space inner products between Gaussian likelihood functions, which we term Gaussian meta-embeddings (GMEs). Meta-embedding extractors can be generatively or discriminatively trained. GMEs extracted by GPLDA have fixed precisions and do not propagate uncertainty. We show that a generalization to heavy-tailed PLDA gives GMEs with variable precisions, which do propagate uncertainty. Experiments on NIST SRE 2010 and 2016 show that the proposed method applied to i-vectors without length normalization is up to 20% more accurate than GPLDA applied to length-normalized i-vectors.

1. Introduction

Embeddings are familiar in modern machine learning. Neural nets to extract word embeddings[1] were already proposed in 2000 by Bengio [1]. Now embeddings are used more generally, for example in state-of-the-art face recognition, e.g. Facenet [2].

Embeddings are becoming popular also in speech and speaker recognition. At Interspeech 2017, eighteen papers had the word 'embedding' in the title.[2] In speaker recognition and spoken language recognition, we have been using *i-vectors*—embeddings extracted by a generative model—for almost a decade [3, 4, 5]. More general embeddings extracted by discriminatively trained DNNs are now appearing in speaker recognition, see for example the *OK Google* system [6], the *Voxceleb* paper [7] and JHU's *x-vectors* [8, 9, 10, 11, 12, 13, 14]. Similar embeddings are also being used for spoken language recognition [15, 16, 17].

Input patterns (sequences of acoustic feature vectors, images, text, ...) live in large, complex spaces, where probability distributions and geometric concepts such as distance are difficult to formulate. The idea with embeddings is that they are representations of complex input patterns that live in simpler spaces, e.g. $\mathbb{R}^d$ (multidimensional Euclidean space), where distance is naturally defined and can be put to work to compare patterns.

At the Johns Hopkins HLTCOE SCALE 2017 Workshop[3] the ongoing research on embeddings for speaker recognition [10] inspired the generalization to *meta-embeddings*. The bulk of the work on meta-embeddings remains unpublished, but a current draft of that work can be followed on GitHub [18].

Traditional embeddings can be interpreted as point estimates for hidden variables of interest and they typically live in low-dimensional Euclidean spaces, where comparisons between them are based on ordinary dot products. Meta-embeddings are *likelihood functions* for those hidden variables. They (meta-embeddings) typically live in infinite-dimensional Hilbert function spaces and comparisons between them are based on more generally defined inner products. This is a considerable generalization, which provides many new opportunities, but also complex challenges, both theoretical and computational. In this work, we restrict attention to multivariate Gaussian likelihood functions, for which the required inner products can be evaluated in closed form.

In future we hope to apply meta-embeddings in speaker recognition in a similar way to i-vectors or x-vectors, in the sense that they will be extracted from the acoustic feature vectors (MFCCs). We regard the work in this paper as a warm-up exercise and a proof on concept, in which we use i-vectors, rather than MFCCs as input. With i-vector inputs, we can profit from simple generative models (Gaussian or heavy-tailed PLDA) that provide elegant closed-form formulas for extracting meta-embeddings.

2. Motivation

In speaker recognition, we already have generative (i-vector) and discriminative (x-vector) embeddings that represent the state of the art and beyond. What additional advantages could we expect from meta-embeddings? The main motivation is to re-design from the ground up a vehicle for the propagation of uncertainty.

The generative i-vector extractor model [4] provides a natural measure of uncertainty in the form of the i-vector posterior. In standard i-vector scoring recipes (PLDA [19] and cosine scoring [3]), only the expected value of that posterior is retained, while the precision (inverse covariance) is discarded. Work to propagate this source of uncertainty through PLDA has had limited success [20, 21, 22]. According to Patrick Kenny [23], i-vector uncertainty propagation does not do what

[1] en.wikipedia.org/wiki/Word_embedding

[2] www.interspeech2017.org/program/technical-program/.

[3] http://hltcoe.jhu.edu/research/scale/scale-2017

10.21437/Odyssey.2018-49

it is supposed to do, but instead gives some benefit as a channel compensator. We speculate that simplifying modelling assumptions and the mean-field variational Bayes approximation (which is required to make the i-vector extractor tractable) may be factors contributing to this problem.

To our knowledge, publications on discriminatively extracted embeddings for speaker recognition have not yet addressed the issue of uncertainty propagation. [4]

As we show below, meta-embeddings, whether extracted discriminatively or generatively, are designed to propagate uncertainty. Let us motivate this in a more general pattern recognition context. Quantifying the uncertainty is very important if a pattern recognizer is to be applicable to variable and sometimes challenging conditions. In speaker recognition, a short, noisy, narrow-band recording should leave much more uncertainty about the speaker than a long, clean, wideband recording. In face recognition, compare a well-lit, high resolution, full-frontal face image to a grainy, low resolution, partly occluded face. In fingerprint recognition, compare a clean, high-resolution ten-print, to a single, distorted, smudged fingermark retrieved from a crime scene.

3. Meta-embeddings

For an elaborate tutorial introduction to meta-embeddings, the interested reader is encouraged to read the first four chapters of [18]. Our summary here is limited to a few paragraphs.

In very general terms, we can describe speaker recognition as the problem of partitioning sets of recordings according to speaker [24]. Set sizes can vary from entire databases to *binary trials* that contain just a pair of recordings. For simplicity we assume: each recording has a single speaker; recordings from different speakers are independent; and the recordings of a given speaker are exchangeable. By De Finetti's theorem [25], exchangeability is equivalent to the concept of the *hidden speaker identity variable*, which is familiar in speaker recognition thanks to the work of Patrick Kenny in JFA [26] and PLDA [19]. Meta-embeddings are *likelihood functions* for the speaker identity variable, of the form:

$$f(\mathbf{z}) \propto P(r \mid \mathbf{z})$$

where $\mathbf{z}$ is the identity variable and r denotes some representation of a recording, e.g. raw speech, MFCCs, i-vector, etc.

For traditional embeddings, the intuitive idea is to retain in the output representation as much as possible of the relevant information that was present in the input, r. For meta-embeddings, that idea is formalized: by definition *all* of the relevant information about the speaker of r must be present in the likelihood function of that recording. (Keep in mind that information content has meaning only relative to some probability model [27] and that for a poor likelihood model, asserting we have all of the relevant information won't do us any good in practice! Just as in any other probabilistic machine learning task, we need to choose the likelihood models wisely and find ways of training the parameters of these models.)

Let $\mathcal{R} = \{r_j\}_{j=1}^n$ denote a set of n recordings. [5] In this paper, we let $\mathbf{z} \in \mathbb{R}^d$ denote a d-dimensional hidden speaker identity variable, for which we assign the standard normal distribution as prior: $\pi(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$. The *meta-embedding*, f_j, extracted from input r_j is the likelihood function:

$$f_j(\mathbf{z}) = k_j P(r_j \mid \mathbf{z}) \tag{1}$$

where $k_j > 0$ is an arbitrary constant that may in general depend on r_j. Take careful note: the meta-embedding is the *whole function*, f_j, rather than some point estimate that lives in $\mathbb{R}^d$.

Let A, B represent two different hypotheses of how $\mathcal{R}$ might be partitioned w.r.t. speaker. Partition A has m hypothesized speakers, indexed by the subsets, $\{\mathcal{S}_i\}_{i=1}^m$, where $\mathcal{S}_i \subseteq \{1, \ldots, n\}$. Likewise, B has m' speakers, indexed by $\{\mathcal{S}'_i\}_{i=1}^{m'}$. Using within-speaker exchangeability and between-speaker independence, the likelihood ratio (LR) comparing A to B can be expressed in terms of the meta-embeddings, f_j as:[6]

$$\begin{aligned} \frac{P(\mathcal{R} \mid A)}{P(\mathcal{R} \mid B)} &= \frac{\prod_{i=1}^m \left\langle \prod_{j \in \mathcal{S}_i} P(r_j \mid \mathbf{z}) \right\rangle_\pi}{\prod_{i=1}^{m'} \left\langle \prod_{j \in \mathcal{S}'_i} P(r_j \mid \mathbf{z}) \right\rangle_\pi} \\ &= \frac{\prod_{i=1}^m \left\langle \prod_{j \in \mathcal{S}_i} f_j(\mathbf{z}) \right\rangle_\pi}{\prod_{i=1}^{m'} \left\langle \prod_{j \in \mathcal{S}'_i} f_j(\mathbf{z}) \right\rangle_\pi} \end{aligned} \tag{2}$$

where the triangle brackets denote expectation w.r.t. $\pi(\mathbf{z})$. The arbitrary scaling constants, $\{k_j\}_{j=1}^n$, are the same in the numerator and denominator and cancel. This equation is very general: any speaker recognition problem that can be formulated in terms of partitions can be expressed purely in terms of the meta-embeddings. This illustrates the principle that the likelihoods represent all the relevant information in the inputs.

If we look only at the first line of (2), we might conclude that to recognize speakers with a principled probabilistic model, we would always need a generative model, which requires possibly complex probability distributions for the observed data, of the form $P(r_j \mid \mathbf{z})$. Keeping in mind that $f_j(\mathbf{z})$ is essentially an un-normalized posterior for $\mathbf{z}$:

$$P(\mathbf{z} \mid r_j) \propto \pi(\mathbf{z}) f_j(\mathbf{z})$$

the last line of (2) shows however that we require only much simpler distributions for $\mathbf{z} \in \mathbb{R}^d$.

Although we *can* extract meta-embeddings from generative models (as we do below), the generative models are by no means required. The RHS of (2) shows that we can score—and therefore also train—meta-embedding systems in purely discriminative ways, without requiring complex generative models for the input data.

> *This paves the way for discriminatively trained, DNN-based speaker recognizers, with principled uncertainty propagation.*

3.1. LR, inner product and pooling

For a binary trial, when $\mathcal{R} = \{r_1, r_2\}$, there are only two possible hypotheses, H_1: there is one speaker; and H_2: there are two speakers. Let f_1 and f_2 denote the meta-embeddings extracted from r_1, r_2, as defined by (1). The likelihood ratio (2) simplifies to:

[4] Although the discriminative x-vector extractor does make use of standard deviations in its temporal pooling stage [10], the uncertainty thus captured is not propagated through to the subsequent PLDA scoring backend.

[5] In this paper, the r_j are i-vectors, but in future work they will be sequences of MFCCs.

[6] This formula is a variation of the principle of *Q-scoring*, which we have introduced in [28, 16].

$$\frac{P(\mathcal{R} \mid H_1)}{P(\mathcal{R} \mid H_2)} = \frac{\big\langle f_1(\mathbf{z}) f_2(\mathbf{z}) \big\rangle_\pi}{\big\langle f_1(\mathbf{z}) \big\rangle_\pi \big\langle f_2(\mathbf{z}) \big\rangle_\pi} = \frac{\big\langle f_1, f_2 \big\rangle}{\big\langle f_1, \mathbf{1} \big\rangle \big\langle f_2, \mathbf{1} \big\rangle} \tag{3}$$

where we have defined the constant function, $\mathbf{1}(\mathbf{z}) = 1$ as well as the *inner product* between two meta-embeddings as:

$$\big\langle f_1, f_2 \big\rangle = \int_{\mathbb{R}^d} f_1(\mathbf{z}) f_2(\mathbf{z}) \pi(\mathbf{z}) d\mathbf{z} \tag{4}$$

We further need the concept of *pooling*. Let $\{r_j\}_{j=1}^k$, be a number of recordings all assumed to be from the same speaker and let $\{f_j\}_{j=1}^k$, be the associated meta-embeddings. Then the *pooled meta-embedding* is the product

$$\bar{f}(\mathbf{z}) = \prod_{i=1}^{k} f_i(\mathbf{z}) \tag{5}$$

This is the likelihood function conditioned on those k recordings, which is the un-normalized form of the pooled posterior:

$$P(\mathbf{z} \mid \{r_j\}_{j=1}^k) \propto \pi(\mathbf{z}) \bar{f}(\mathbf{z}) \tag{6}$$

As shown in [18], all likelihood ratios of the form (2) can be expressed in terms of these two primitive operations: pooling and inner products. Since our inner products are expectations of products, we can alternatively let our primitive operations be *pooling and expectation*.

Given some regularity conditions on the likelihood functions, our meta-embeddings live in a Hilbert space, which is a vector space equipped with an inner product. Although this Hilbert space is typically infinite-dimensional, it has a geometry just like Euclidean space. In meta-embedding space, norms, distances and angles are well-defined and have meaningful interpretations and practical applications in scoring and training of speaker recognizers. We lack space here, but the details are in [18].

3.2. Gaussian meta-embeddings

In practice, we need our primitive operations (pooling and expectation) to be computationally tractable. In this paper we restrict attention to multivariate Gaussian likelihood functions. The *Gaussian meta-embedding* (GME), extracted from a recording r_j is defined as:

$$f_j(\mathbf{z}) = \exp\big[\mathbf{a}_j'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}_j\mathbf{z}\big] \tag{7}$$

where f_j is represented by its *natural parameters*: $\mathbf{a}_j \in \mathbb{R}^d$ and the d-by-d positive semi-definite *precision matrix*, $\mathbf{B}_j$. In future work, we envisage a discriminatively trained meta-embedding extractor where a DNN would process r_j (a sequence of MFCCs) and then output $\mathbf{a}_j$ and some sensible representation of $\mathbf{B}_j$. In this paper, as warm-up exercise, we let the r_j be i-vectors and we use the PLDA model to derive relatively simple functions to extract $\mathbf{a}_j, \mathbf{B}_j$.

Since Gaussians are closed w.r.t. products and our representation is essentially logarithmic, pooling is easy: we simply need to add the natural parameters. The toolbox of primitive operations is completed by a closed-form expression for Gaussian expectations. For a (raw or pooled) meta-embedding, f, represented by $\mathbf{a}, \mathbf{B}$, the expectation w.r.t. the prior is [18]:

$$\log E(\mathbf{a}, \mathbf{B}) = \log\big\langle f \big\rangle_\pi = \log \int_{\mathbb{R}^d} \mathbf{a}'\mathbf{z} - \frac{1}{2}\mathbf{z}'(\mathbf{B}+\mathbf{I})\mathbf{z}\, d\mathbf{z} = \frac{1}{2}\mathbf{a}'(\mathbf{B}+\mathbf{I})^{-1}\mathbf{a} - \frac{1}{2}\log|\mathbf{B}+\mathbf{I}| \tag{8}$$

In the general case, Cholesky factorization would be the standard tool for this computation, but the GMEs that we extract in this paper have diagonalizable precisions that allow for much faster computations.

We can combine the pooling and expectation formulas to compute arbitrary likelihood ratios of the form (2). As an example, for binary trials, the LR becomes:

$$\frac{P(\mathcal{R} \mid H_1)}{P(\mathcal{R} \mid H_2)} = \frac{E(\mathbf{a}_1 + \mathbf{a}_2, \mathbf{B}_1 + \mathbf{B}_2)}{E(\mathbf{a}_1, \mathbf{B}_2) E(\mathbf{a}_2, \mathbf{B}_2)} \tag{9}$$

4. PLDA as meta-embedding extractor

In what follows, we let our recording representations be D-dimensional i-vectors: $\mathbf{r}_j \in \mathbb{R}^D$, and we derive a meta-embedding extractor via a heavy-tailed PLDA model. For more details see the chapter on *Generative meta-embeddings* in [18].

The hidden variable is $\mathbf{z} \in \mathbb{R}^d$ and we require: $d < D$. The standard normal prior, $\pi(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$, forms part of our PLDA model. (This is a simplification of Kenny's original model, which had a heavy-tailed prior [19].) The PLDA model says that for every speaker, the identity variable $\mathbf{z}$ is sampled independently from $\pi(\mathbf{z})$; and every i-vector, $\mathbf{r}_j$, for a speaker having identity variable $\mathbf{z}$, is generated as:

$$\mathbf{r}_j = \mathbf{F}\mathbf{z} + \boldsymbol{\eta}_j, \qquad \boldsymbol{\eta}_j \sim \mathcal{T}(\mathbf{0}, \mathbf{W}, \nu) \tag{10}$$

where $\mathbf{F}$ is a D-by-d factor loading matrix and where the 'channel' noise, $\boldsymbol{\eta}_j$, is drawn from a multivariate t-distribution [29], having zero mean,[7] precision $\mathbf{W}$ and *degrees of freedom*, $\nu > 0$. The PLDA model parameters are $\mathbf{F}, \mathbf{W}, \nu$. The meta-embedding that this model extracts from $\mathbf{r}_j$ is simply:

$$f_j(\mathbf{z}) \propto \mathcal{T}(\mathbf{r}_j \mid \mathbf{F}\mathbf{z}, \mathbf{W}, \nu) \tag{11}$$

This is a t-distribution for $\mathbf{r}_j$, but what does it look like as a function of $\mathbf{z}$? Provided that $D > d$ and $\mathbf{F}'\mathbf{W}\mathbf{F}$ is invertible, it is shown in [18] that it is another t-distribution, with *increased* degrees of freedom, $\nu' = \nu + D - d$

$$f_j(\mathbf{z}) \propto \mathcal{T}(\mathbf{z} \mid \mathbf{B}_j^{-1}\mathbf{a}_j, \mathbf{B}_j, \nu') \tag{12}$$

where

$$\mathbf{B}_j = b_j \bar{\mathbf{B}}, \quad \mathbf{a}_j = b_j \mathbf{F}'\mathbf{W}\mathbf{r}_j, \quad b_j = \frac{\nu + D - d}{\nu + \mathbf{r}_j'\mathbf{G}\mathbf{r}_j} \tag{13}$$

and

$$\bar{\mathbf{B}} = \mathbf{F}'\mathbf{W}\mathbf{F}, \qquad \mathbf{G} = \mathbf{W} - \mathbf{W}\mathbf{F}\bar{\mathbf{B}}^{-1}\mathbf{F}'\mathbf{W} \tag{14}$$

In a typical PLDA model, we have $d \in [100, 200]$ and $D \in [400, 600]$, so that $\nu' = \nu + D - d$ is large, making the meta-embedding practically Gaussian. We therefore approximate:

$$f_j(\mathbf{z}) \approx \mathcal{N}(\mathbf{z} \mid \mathbf{B}_j^{-1}\mathbf{a}_j, \mathbf{B}_j^{-1}) \propto \exp\big[\mathbf{a}_j'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}_j\mathbf{z}\big] \tag{15}$$

Several comments are in order:

[7] We can let the model have non-zero mean, but in practice it is simpler just to zero the mean of the training data.

- For the heavy-tailed case, with small ν, the meta-embedding precisions, $\mathbf{B}_j = b_j\bar{\mathbf{B}}$, vary as a function of the data. This is the uncertainty propagation.
- Since precisions differ only by a scalar, we can simplify pooling and expectation. For pooling, addition of precisions simplifies to scalar addition. For the expectations (8), we do not have to keep Cholesky factorizing every time. By precomputing an eigenanalysis of $\bar{\mathbf{B}}$, we can diagonalize all $\mathbf{I}+\mathbf{B}_j$, which allows fast scoring and training [18].
- Notice that $\mathbf{GF} = \mathbf{0}$, so that

$$\mathbf{r}_j'\mathbf{G}\mathbf{r}_j = \boldsymbol{\eta}_j'\mathbf{G}\boldsymbol{\eta}_j$$

  is an ancillary statistic, *independent* of the speaker variable $\mathbf{z}$, but nevertheless important for complete inferences about the speaker. The heavy-tailed noise, $\boldsymbol{\eta}_j$ is essentially Gaussian noise with random variance. For a 'bad' i-vector, with a large noise vector, $\mathbf{r}_j'\mathbf{G}\mathbf{r}_j$ will also be large and the precision scaling factor, b_j will be small. For a 'good' i-vector, with small noise, the opposite happens.
- For the more familiar case of Gaussian PLDA, when $\nu \to \infty$, we have $b_j \to 1$ and the meta-embedding precisions remain constant. In this case we cannot exploit the informtation in $\mathbf{r}_j'\mathbf{G}\mathbf{r}_j$.

4.1. Length normalization

In [19] heavy-tailed PLDA was shown to be a better model of i-vectors than Gaussian PLDA, but the computational cost was considerable. Subsequently, [30] showed that the i-vectors could instead be Gaussianized via a simple length normalization procedure. This matched the accuracy of heavy-tailed PLDA, with negligible extra computational cost. The heavy-tailed nature of i-vectors was also addressed in [31], where an iterative scaling procedure was proposed.

In this paper, our approximation (15) has the advantage of a fast, closed-form estimate of the scale factor: $\mathbf{r}_j'\mathbf{G}\mathbf{r}_j$. Of course, length normalization would interfere with that estimate. We therefore apply the proposed Gaussian meta-embeddings (GMEs) to i-vectors *without* length normalization. In our experiments below we compare against Gaussian PLDA applied to i-vectors both with and without length normalization.

4.2. GME extractor and scoring

In summary, the PLDA model (whether Gaussian or heavy-tailed), provides the functional form (13), (14) and (15), for extracting Gaussian meta-embeddings from i-vectors. We shall explore both generative and discriminative methods for training the parameters of this GME extractor.

It is also worth emphasizing the following difference between i-vectors and x-vectors on the one hand, versus meta-embeddings on the other:

- The extractors for i-vectors and x-vectors are typically trained separately from the PLDA scoring backend. (In practice, this has many advantages.)
- Meta-embeddings, once extracted, do not need an additional backend for scoring. The meta-embeddings contain within themselves everything that is needed to produce scores and this is done in general by (2), or more specifically for GMEs applied to binary trials by (9).

Of course, i-vectors and x-vectors *can* be scored in a parameterless backend via cosine scoring, but this is usually less accurate.

In machine learning, e.g. in Facenet [2], the parameterless scoring strategy is preferred. The philosophy there can be summarized as: Train the embedding extractor such that embeddings of the same face are close (in Euclidean distance), while embeddings of different faces are far apart. That brute-force, geometric strategy has been very successful and should be appreciated as such. We do however hope that additional benefits can be eventually reaped as a result of the probabilistic strategy of the meta-embedding design.

Finally, it is worth repeating that comparisons between meta-embeddings *can* be interpreted in terms of a slightly more complex geometry, in terms of angles and norms. The interested reader is referred to the chapter entitled *The structure of meta-embedding space* in [18].

5. Training

In what follows, let $\mathcal{R}$ denote the recordings in the training database; $\mathcal{L}$, the 'labels', or true partition of the database w.r.t. speaker; $\mathbf{Z}$ all of the hidden speaker identity variables (one per speaker); θ the generative model parameters (in this paper $\theta = (\mathbf{W}, \mathbf{F}, \nu)$; and ϕ a set of variational parameters which would be needed for variational Bayes (VB) training strategies.

Training strategies for complex probabilistic models are perhaps best understood via comparison with the celebrated *variational autoencoder* (VAE) [32] as described next.

5.1. Generative training

VB training effects approximate maximization of the marginal likelihood—here $P(\mathcal{R} \mid \mathcal{L}, \theta)$. Classical VB maximizes a lower bound to the marginal likelihood and is applicable to conjugate-exponential models with intractable posteriors and marginal likelihoods [29]. The VAE generalizes this to a wider class of (deep generative) models by using a *stochastic approximation* of the VB lower bound [32]. The VAE has two parts, decoder and encoder. The decoder is the generative model likelihood, $P(\mathcal{R}, \mid \mathcal{L}, \mathbf{Z}, \theta)$. The encoder is a tractable variational posterior, $Q(\mathbf{Z} \mid \mathcal{R}, \mathcal{L}, \phi)$, which approximates the true intractable posterior, $P(\mathbf{Z} \mid \mathcal{R}, \mathcal{L}, \theta)$. VAE training is accomplished by maximization of the stochastic VB lower bound w.r.t. both decoder and encoder parameters, θ and ϕ.

In this paper, we did not use VAE. Instead we used a crude shortcut. We trained a Gaussian PLDA model with the usual EM-algorithm and then simply plugged in a hand-selected value for ν to make the model heavy-tailed.

Future work may however investigate VAE as a more powerful generative training strategy. The heavy-tailed PLDA model of section 4 has intractable posteriors, $P(\mathbf{z} \mid r_j)$, formed by the product of the Gaussian prior and the t-distribution likelihoods. As we pointed out however, the t-distribution likelihood is almost Gaussian, so that Gaussian variational posteriors can be expected to work well. Unfortunately, the VB lower bound (Gaussian expectations of logarithms of t-distributions) does not have a closed form, so that VAE rather than classical VB would be necessary. For this model, the variational parameters could be tied to the generative parameters, using (13) and (14). However, VB allows unconstrained optimization of the variational parameters, which could give advantages in both accuracy and computational complexity.

Alternatively, another solution for training is the classical mean-field VB solution for heavy-tailed PLDA of [19], where the channel noise scaling factors are treated as hidden variables.

5.2. Discriminative training

VAE training might be a good idea for simpler models such as heavy-tailed PLDA applied to i-vectors. However, for more complex models applied to acoustic features, discriminative training starts to look more attractive.

- For VAE training, we have a complexity doubling effect—we have to build a complex encoder, a complex decoder and also manage the non-trivial interface between them—see for example [28]. Once training is complete, the decoder is no longer needed for runtime scoring.
- In discriminative training, no decoder is needed and we only need to train the equivalent of the encoder.

In future work, for more complex models that extract meta-embeddings from acoustic features, we envisage that purely discriminative training methods could provide an easier route to success.

Binary cross-entropy (BXE), applied to pairs of recordings, is a popular discriminative training criterion in speaker recognition [33, 34, 35, 6, 8]. BXE can indeed also be used to train meta-embedding extractors and that is what we do in this paper. The scoring formula needed during BXE training of the GME extractor is (9).[8]

For future work, we note that BXE is by no means the only option—see [18] for a variety of other proposed discriminative training criteria. In particular, we would like to highlight the computationally attractive *pseudolikelihood* criterion, which does not rely on a quadratic expansion into binary trials. In addition, pseudolikelihood is a *proper scoring rule* for this training problem in a stricter sense than BXE and may give advantages as a calibration-sensitive training and evaluation criterion.

6. Experiments

6.1. I-vector extraction

In this paper our recordings are represented by i-vectors. For all of the experiments we use a single database of i-vectors, extracted as described below.

We used 60-dimensional spectral features: 20 MFCCs, including C_0, augmented with Δ and $\Delta\Delta$. The features were short-term mean and variance normalized over a 3 second sliding window. With those features, we train a GMM UBM with 2048 diagonal components in gender independent fashion. Then, we collect sufficient statistics and train the i-vector extractor, with 600-dimensional i-vectors. These i-vectors serve us as input to either the PLDA baseline or to the new meta-embedding extractor. In both cases, i-vectors are transformed with global mean normalization. Then, for one of the baseline PLDA systems (baseline 1) we also apply length normalization. The second PLDA system (baseline 2) is applied to i-vectors without length normalization.

6.2. Datasets and evaluation metrics

UBM, i-vector extractor and PLDA models are trained on the PRISM dataset [37], containing Fisher parts 1 and 2, Switchboard 2, 3 and Switchboard cellphone phases. Also, NIST SRE 2004–2008 (from the MIXER collections) are added to the training. In total, the set contains approximately 100K utterances coming from 16241 speakers. We used 8000 randomly selected files for UBM training and full set to train i-vector extractor and PLDA.

For training the GME extractor we use the same training list. However, for discriminative training via stochastic gradient descent (SGD), we split the set into training and cross-validation subsets. Cross validation was done on a randomly selected subset of 10% of the speakers, leaving the other 90% for training. This gave 8740 utterances for cross validation and 90309 for training.

We evaluate performance on the female part of NIST SRE 2010, condition 5, which consists of English telephone data [38]. Additionally, we report the results on the NIST SRE 2016 evaluation set (both males and females). We report the results on the whole SRE'16 and also separately for each of the two language subsets, Cantonese and Tagalog. As evaluation metrics, we use the equal error rate (EER, in %) as well as the average minimum detection cost function for two operating points ($C_{\min}^{\text{Prm}}$). The two operating points are the ones of interest in the NIST SRE 2016 [39], namely the probability of target trials 0.01 and 0.005.

6.3. GME extractor initialized from PLDA

As mentioned in section 4, the GME extractor defined by (13) with $\nu \to \infty$ is equivalent to Gaussian PLDA. In this case, we can set the parameters of the GME extractor in such a way that log-likelihood scores computed with meta-embeddings will be equal to the scores provided by Gaussian PLDA. To see this, recall that in Gaussian PLDA the log-likelihood-ratio score for two i-vectors, $\mathbf{r}_1$ and $\mathbf{r}_2$, can be expressed as

$$S_{\text{PLDA}} = 2\mathbf{r}_1'\mathbf{\Lambda}\mathbf{r}_2 + \mathbf{r}_1'\mathbf{\Gamma}\mathbf{r}_1 + \mathbf{r}_2'\mathbf{\Gamma}\mathbf{r}_2 + (\mathbf{r}_1+\mathbf{r}_2)'\mathbf{c} + k, \quad (16)$$

where the parameters $\mathbf{\Lambda}$, $\mathbf{\Gamma}$, $\mathbf{c}$ and k are calculated from the parameters of the PLDA model (see eq. (8) of [34]). Since we subtracted the global mean from the i-vectors, we have $\mathbf{c} = \mathbf{0}$, so that the linear term can be omitted.

Substituting for $\mathbf{a}$ and $\mathbf{B}$ in (8) with expressions given by (13) and setting $b_j = 1$, the GME log-likelihood ratio score is:

$$\begin{aligned} S_{\text{GME}} = &\frac{1}{2}\mathbf{r}_1'\mathbf{W}'\mathbf{F}((\mathbf{I}+2\bar{\mathbf{B}})^{-1} - (\mathbf{I}+\bar{\mathbf{B}})^{-1})\mathbf{F}'\mathbf{W}\mathbf{r}_1 \\ &+ \frac{1}{2}\mathbf{r}_2'\mathbf{W}'\mathbf{F}((\mathbf{I}+2\bar{\mathbf{B}})^{-1} - (\mathbf{I}+\bar{\mathbf{B}})^{-1})\mathbf{F}'\mathbf{W}\mathbf{r}_2 \\ &+ \mathbf{r}_1'\mathbf{W}'\mathbf{F}((\mathbf{I}+2\bar{\mathbf{B}})^{-1})\mathbf{F}'\mathbf{W}\mathbf{r}_2 \\ &- \frac{1}{2}\log|\mathbf{I}+2\bar{\mathbf{B}}| + \log|\mathbf{I}+\bar{\mathbf{B}}| \end{aligned} \quad (17)$$

Now, comparing (16) and (17), we see that $S_{\text{GME}} = S_{\text{PLDA}}$ when

$$\begin{aligned} \mathbf{\Gamma} &= \frac{1}{2}\mathbf{W}'\mathbf{F}((\mathbf{I}+2\bar{\mathbf{B}})^{-1} - (\mathbf{I}+\bar{\mathbf{B}})^{-1})\mathbf{F}'\mathbf{W}, \\ \mathbf{\Lambda} &= \frac{1}{2}\mathbf{W}'\mathbf{F}((\mathbf{I}+2\bar{\mathbf{B}})^{-1})\mathbf{F}'\mathbf{W}, \\ k &= -\frac{1}{2}\log|\mathbf{I}+2\bar{\mathbf{B}}| + \log|\mathbf{I}+\bar{\mathbf{B}}|. \end{aligned} \quad (18)$$

Our GME extractor was initialized by solving for $\mathbf{F}$ and $\mathbf{W}$, while various values for ν were plugged in by hand. In the heavy-tailed regime (small ν), our results are not very sensitive to the exact value—we report results for $\nu = 2$. As a sanity check, we also tried $\nu \to \infty$ to verify the equivalence with Gaussian PLDA. This is achieved by setting the $b_j = 1$.

[8] Although we did not do that experiment here, notice that discriminative training of the GME extractor with $\nu \to \infty$ is equivalent to discriminatively trained Gaussian PLDA [34, 36].

In our experiments below, we try the generatively initialized GME extractor as-is, as well as a discriminatively trained extractor as described next.

6.4. Discriminative GME extractor training

As mentioned in section 5.2 we used binary cross-entropy (BXE) scored on pairs of i-vectors to discriminatively train the parameters of the GME extractor. We used the initialization from Gaussian PLDA as described above, with ν plugged in, followed by minibatch stochastic gradient descent (SGD) on the BXE objective.

The (rather large) minibatches are formed as follows. We randomly select[9] two sets of 5000 i-vectors each from the training data to serve as enrollment and test sets. Then, all i-vectors in the first set are scored against all in the second. We do not filter out any trials except for the cases where i-vectors are scored against themselves. As expected, the amount of non-target trials in a batch is much higher than target trials and the ratio can vary between the batches. To compensate for this, we separately compute BXE for target and non-target examples in the current batch and re-weight them so have an effective ratio of 400 non-targets for every 3 targets.

The extractor parameters, $\mathbf{W}$ and $\mathbf{F}$, are updated by back-propagating gradients through the BXE objective, through the scoring formula (9) and the extractor formula (13) and (14). The value of ν remains fixed at the plugged in value throughout training. Training continues until the BXE objective stops improving on the held out cross-validation set.

6.5. Multi-enroll trials

The SRE'16 evaluation set includes some trials with multiple enrollment recordings. For the results reported here, we took the shortcut of simply averaging enrollment i-vectors. Of course, both PLDA and meta-embeddings provide for more principled enrollment pooling and that will be explored in future.

6.6. Results

Table 1 compares the results for the two Gaussian PLDA baselines against three variants of Gaussian meta-embeddings.

The first part of the table shows the results of our two baselines. Baseline 1 is Gaussian PLDA with $D = 600$ and $d = 200$, applied to length-normalized i-vectors. Baseline 2 is the same, but without length normalization. As usual, length normalization helps a lot—it makes the data more Gaussian to better fit the Gaussian PLDA model.

The second part of the table shows results for three GME configurations, all of them applied to i-vectors *without* length normalization. The first GME result is the sanity check that shows the equivalence between Baseline 2 and the GME extractor initialized from it, with $\nu \to \infty$. The second GME result is the same as the previous one (PLDA initialization, no further training) but now with $\nu = 2$. Notice that this already does better in all cases than Gaussian PLDA without length normalization. Finally the third GME result shows that after further discriminative training, *GME without length normalization can do better than Gaussian PLDA with length normalization*.

By changing the degrees of freedom parameter, ν, we have effectively relaxed the Gaussian modelling assumptions. In our experiments, we have tried several different values for ν. We found that parameter ν can vary in a wide range of values where all of them provide similar performance. Here, we picked $\nu = 2$. As results indicate, the training not only mitigates the degradation brought by the lack of length normalization but even brings further improvements compared to both baselines in most of the cases.

[9] with replacement

6.7. Computational complexity

In our experiments, starting from i-vectors, Gaussian PLDA scoring of the whole SRE'10 and SRE'16 evaluation sets required respectively $0.4s$ and $1.5s$ in wall clock time. The GME solution required roughly double the time for the same tasks. We did not have an implementation of Kenny's heavy-tailed PLDA [19] to hand for direct comparison, but we do know that its computational complexity has thus far prevented it from being widely adopted.

7. Discussion

This paper introduces *meta-embeddings*, which are intended as a future alternative to i-vectors and x-vectors in speaker recognition, and indeed in other areas of machine learning as an alternative to traditional embeddings. The chief motivation for meta-embeddings is to build discriminatively trainable recognizers that allow the principled propagation of uncertainty, all the way from the input to the final output. We expect these advantages to be most noticeable in applications with varying and sometimes challenging quality of the inputs.

We do not yet have a full meta-embedding replacement for i-vectors or x-vectors, but were able to demonstrate the utility of our new design principles by creating a new i-vector scoring backend that is more accurate than the long-standing state of the art represented by length normalization and Gaussian PLDA. For SRE'10, we showed a 20% relative improvement in EER and for SRE'16 a 1% absolute improvement. This improvement was achieved purely by replacing the backend—without resorting to data augmentation, fusion, domain adaptation, or score normalization.

Our ongoing work on meta-embeddings can be followed on GitHub at [18].

7.1. The shrinkage problem

Although our results show that heavy-tailed PLDA performs better than Gaussian PLDA on i-vectors without length normalization, there remains a problem with i-vectors extracted from recordings with short durations. In the usual i-vector extractor, the effect of the standard normal prior is to shrink short-duration i-vectors towards the origin. The heavy-tailed PLDA model breaks down in such cases, because when $\mathbf{r}_j'\mathbf{G}\mathbf{r}_j$ decreases, it extracts meta-embeddings with *higher* precision. This is the opposite of what we want—shorter durations should give more uncertainty, not less. This inconsistency can be explained as follows. In an ideal world, the whole PLDA model should form the prior for the i-vector extractor. This is however not practical and we are forced to compromise by using the simpler standard normal prior instead.

We conjecture that the π-vectors of [40] might help to mitigate this problem. The π-vectors are extracted similarly to i-vectors, but without the regularizing prior and do not shrink towards zero for short durations. Another way to see it is that π-vectors are point-estimates extracted from the i-vector likelihood function, rather than from the i-vector posterior. The likelihood function is uncontaminated by the inaccurate prior.

This is however not a completely satisfactory solution,

Table 1: Comparison of accuracies on SRE 2010 and 2016 of Gaussian PLDA with length normalization (baseline 1) and without it (baseline 2), versus GME (without length normalization). No training means initialized from baseline. Retraining is discriminative.

System	SRE10 c05,f		SRE16, all		SRE16, Cantonese		SRE16, Tagalog	
	$C_{\min}^{\text{Prm}}$	EER	$C_{\min}^{\text{Prm}}$	EER	$C_{\min}^{\text{Prm}}$	EER	$C_{\min}^{\text{Prm}}$	EER
baseline 1	0.262	2.54	0.959	16.66	**0.684**	9.80	0.983	21.53
baseline 2	0.329	4.21	0.963	19.13	0.726	13.03	0.985	23.10
GME $\nu \to \infty$, no training	0.329	4.21	0.963	19.13	0.726	13.03	0.985	23.10
GME $\nu = 2$, no training	0.299	2.87	0.960	16.99	0.692	10.03	0.979	21.63
GME $\nu = 2$, retrained	**0.213**	**2.05**	**0.879**	**15.56**	0.734	**9.51**	**0.955**	**20.66**

because we are still going via a point-estimate, which does not properly convey the uncertainty inherent in short-duration recordings. As mentioned, our next goal is to construct meta-embedding extractors that work directly on the acoustic features, rather than via i-vector-like point-estimates.

8. FAQ for future research

Q: I have an existing DNN that extracts x-vectors from MFCCs, followed by a Gaussian PLDA backend. How do I generalize this to a meta-embedding extractor?

A: One solution is to replace your PLDA with our new GME backend. Replace our $\mathbf{r}_j$ with your x-vectors. Initialize the backend from PLDA (with $D \gg d$) as we did. It is probably not worthwhile learning ν, just fix it to say 1 or 2. Discriminative training of the GME backend (with fixed extractor) may already improve accuracy, as it did for us. The next step is to *jointly* optimize the extractor and the backend. This should encourage the uncertainty to be propagated from the extractor to the backend, through the $(D-d)$-dimensional complement of the speaker subspace.

A: Variants of the above recipe, where the interface between the original extractor and the back-end is simplified may ultimately give better solutions, but they may be more difficult to initialize. For example, one could replace (13) and (14) as follows. Split your D-dimensional x-vector into two parts: say $\mathbf{x}_j \in \mathbb{R}^d$ and $\mathbf{y}_j \in \mathbb{R}^{D-d}$. Then do $b_j = \frac{\nu+D-d}{\nu+\mathbf{y}'\mathbf{y}}$, $\mathbf{a}_j = b_j\mathbf{x}_j$ and $\mathbf{B}_j = b_j\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is a trainable diagonal matrix with positive entries.

Q: Which discriminative training criterion should I use?

A: In our experience, the multiclass cross-entropy that is currently used to train the Kaldi x-vector extractors of [10] and [11] will not work for training the GME backend, and by implication also not for the joint training stage. Do use your existing method to pre-train your extractor, but then for the further training, change the criterion. The first option to try is BXE as used in this paper. We were not successful in using BXE to train from random initialization, but it did work after PLDA initialization. A look at pseudolikelihood, or some of the other criteria proposed in [18] may be worthwhile.

Q: Are there any important tricks that are not mentioned in the paper?

A: Yes. For example, to simplify the backend, set $\mathbf{W} = \mathbf{I}$ and learn a linear transform of the $\mathbf{r}_j$ instead. Constrain (or coerce with a suitable L2 regularization penalty) $\bar{\mathbf{B}} = \mathbf{F}'\mathbf{F}$ to be diagonal. We would be happy to assist with further details.

A: The generative heavy-tailed PLDA model can be used to generate synthetic data, with known properties. We found this invaluable in experiments to explore various discriminative training criteria.

9. Acknowledgements

This work was started at the Johns Hopkins University HLTCOE SCALE 2017 Workshop. The authors thank the workshop organizers for inviting us to attend and (in the case of Niko Brümmer) for generous travel funding. Themos Stafylakis has been funded by the European Commission program Horizon 2020, under grant agreement no.706668 (Talking Heads). The work was also supported by Czech Ministry of Interior project No. VI20152020025 "DRAPAK" Google Faculty Research Award program, Technology Agency of the Czech Republic project No. TJ01000208 "NOSICI", and by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

10. References

[1] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *NIPS*, 2000.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015. [Online]. Available: http://arxiv.org/abs/1503.03832

[3] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Interspeech*, Brighton, UK, September 2009.

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

[5] D. Martínez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," in *Interspeech*, 2011.

[6] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*, 2016. [Online]. Available: http://arxiv.org/abs/1509.08062

[7] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Interspeech*, 2017. [Online]. Available: https://arxiv.org/abs/1706.08612

[8] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verifi-

cation," in *IEEE Workshop on Spoken Language Technology*, 2016.

[9] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Interspeech*, Stockholm, 2017.

[10] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*, Calgary, 2018.

[11] M. McLaren, D. Castan, M. K. Nandwana, L. Ferrer, and E. Yılmaz, "How to train your speaker embeddings extractor," in *Odyssey: The Speaker and Language Recognition Workshop*, Les Sables d'Olonne, 2018, submitted.

[12] H. Bredin and T. March, "Triplet loss for speaker turn embedding," in *ICASSP*, 2017.

[13] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," *arXiv*, 2017. [Online]. Available: arxiv.org/abs/1705.02304

[14] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in *Interspeech*, 2017.

[15] G. Gelly and J. L. Gauvain, "Spoken language identification using LSTM-based angular proximity," in *Interspeech*, Stockholm, 2017.

[16] J. Villalba, N. Brümmer, and N. Dehak, "End-to-end versus embedding neural networks for language recognition in mismatched conditions," in *Odyssey: The Speaker and Language Recognition Workshop*, Les Sables d'Olonne, 2018, submitted.

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

[18] N. Brümmer, L. Burget, P. Garcia, O. Plchot, J. Rohdin, D. Romero, D. Snyder, T. Stafylakis, A. Swart, and J. Villalba, "Meta-embeddings: a probabilistic generalization of embeddings in machine learning," In progress. Draft available: github.com/bsxfan/meta-embeddings, 2017-2018.

[19] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010, keynote presentation.

[20] S. Cumani, O. Plchot, and P. Laface, "On the use of i-vector posterior distributions in PLDA," *IEEE Trans. ASLP*, vol. 22, no. 4, 2014.

[21] P. Kenny, T. Stafylakis, P. Ouellet, M. J. Alam, and P. Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *IEEE ICASSP*, 2013.

[22] T. Stafylakis, P. Kenny, P. Ouellet, J. Perez, M. Kockmann, and P. Dumouchel, "Text-dependent speaker recognition using PLDA with uncertainty propagation," in *Interspeech*, 2013.

[23] P. Kenny, T. Stafylakis, J. Alam, V. Gupta, and M. Kockmann, "Uncertainty modeling without subspace methods for text-dependent speaker recognition," in *Speaker Odyssey: The Speaker and Language Recognition Workshop*, Bilbao, 2016.

[24] N. Brümmer and E. de Villiers, "The speaker partitioning problem," in *Odyssey Speaker and Language Recognition Workshop*, Brno, Czech Republic, June 2010.

[25] Y. S. Chow and H. Teicher, *Probability theory: Independence, interchangeability, martingales*, 3rd ed., ser. Springer Texts in Statistics. New York: Springer, 1997.

[26] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," CRIM, Montreal, Tech. Rep. CRIM-06/08-13, 2005.

[27] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[28] J. Villalba, N. Brümmer, and N. Dehak, "Tied variational autoencoder backends for i-vector speaker recognition," in *Interspeech*, Stockholm, 2017.

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

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

[31] S. Cumani and P. Laface, "I-vector transformation and scaling for PLDA based speaker recognition," in *Odyssey: The Speaker and Language Recognition Workshop*, Bilbao, 2016.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[33] N. Brümmer, L. Burget, J. Černocky, O. Glembek, F. Grézl, M. Karafíat, D. A. van Leeuwen, P. Matějka, P. Schwarz, and A. Strasheim, "Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST Speaker Recognition Evaluation 2006," *IEEE TASLP*, vol. 15, no. 7, September 2007.

[34] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matějka, and N. Brümmer, "Discriminatively trained probabilistic linear discriminant analysis for speaker verification," in *ICASSP*, Prague, CZ, May 2011.

[35] N. Brümmer, A. Swart, and D. van Leeuwen, "A comparison of linear and non-linear calibrations for speaker recognition," in *Odyssey 2014: The Speaker and Language Recognition Workshop*, Joensuu, 2014. [Online]. Available: http://arxiv.org/abs/1402.2447

[36] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, "Pairwise discriminative speaker verification in the i-vector space," *IEEE Transactions on audio, speech and language processing*, vol. 21, no. 6, pp. 1217–1227, 2013.

[37] L. Ferrer, H. Bratt, L. Burget, H. Cernocky, O. Glembek, M. Graciarena, A. Lawson, Y. Lei, P. Matejka, O. Plchot *et al.*, "Promoting robustness for speaker modeling in the community: the prism evaluation set," *https://code.google.com/p/prism-set/*, 2012.

[38] NIST, "The nist year 2010 speaker recognition evaluation plan," www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf, 2010.

[39] "The 2016 NIST speaker recognition evaluation plan (sre16)," https://www.nist.gov/file/325336.

[40] D. Garcia-Romero and A. McCree, "Subspace-constrained supervector PLDA for speaker verification," in *Interspeech*, Lyon, France, 2013.