Interspeech 2018
2-6 September 2018, Hyderabad

Fast variational Bayes for heavy-tailed PLDA applied to i-vectors and x-vectors

Anna Silnova[1], *Niko Brümmer*[2], *Daniel Garcia-Romero*[3], *David Snyder*[3], *and Lukáš Burget*[1]

[1]Brno University of Technology, Czech Republic
[2]Nuance Communications, South Africa
[3]Johns Hopkins HLTCOE, USA

{isilnova,burget}@fit.vutbr.za, niko.brummer@gmail.com, dgromero@jhu.edu

Abstract

The standard state-of-the-art backend for text-independent speaker recognizers that use i-vectors or x-vectors, is Gaussian PLDA (G-PLDA), assisted by a Gaussianization step involving length normalization. G-PLDA can be trained with both generative or discriminative methods. It has long been known that heavy-tailed PLDA (HT-PLDA), applied without length normalization, gives similar accuracy, but at considerable extra computational cost. We have recently introduced a fast scoring algorithm for a discriminatively trained HT-PLDA backend. This paper extends that work by introducing a fast, variational Bayes, generative training algorithm. We compare old and new backends, with and without length-normalization, with i-vectors and x-vectors, on SRE'10, SRE'16 and SITW.
Index Terms: speaker recognition, variational Bayes, heavy-tailed PLDA

1. Introduction

We extend our previous work in [1], where we did discriminative training of a heavy-tailed PLDA model (HT-PLDA), applied to i-vectors. In this paper, we explore instead a generative training solution and we apply it to both i-vectors [2] and x-vectors [3]. The advantage of the generative training is that it is orders of magnitude faster than the discriminative one.

In [4] HT-PLDA was shown to be a better model of i-vectors than Gaussian PLDA (G-PLDA), but the computational cost was considerable. Subsequently, [5] showed that the i-vectors could instead be Gaussianized via a simple length normalization procedure. This matched the accuracy of HT-PLDA, with negligible extra computational cost and established G-PLDA with length-normalization as the standard backend for scoring text-independent i-vector speaker recognizers. Recently, the same scoring recipe was applied also to x-vectors [3, 6, 7].

In this paper, we revisit HT-PLDA, using a slightly simplified model for which we present fast training and scoring algorithms, with speed comparable to G-PLDA. The HT-PLDA model can be applied without length normalization. We demonstrate accuracy gains on some data sets, including EER=2.7% on SITW and EER=3.2% on SRE'16 Cantonese. Since we have effectively removed the computational impediment, we encourage other researchers to experiment with this backend as an alternative. We make open source code available to facilitate such experiments.

2. HT-PLDA model

The generative HT-PLDA model is shown in graphical model notation [8] in figure 1 and is defined as follows. For every speaker, i, let all of the available observations of that speaker (N_i of them) be denoted as $\mathcal{R}_i = \{\mathbf{r}_{ij}\}_{j=1}^{N_i}$, where the $\mathbf{r}_{ij} \in \mathbb{R}^D$ are i-vectors, or x-vectors, of dimension D. For every speaker, a hidden speaker identity variable, $\mathbf{z}_i \in \mathbb{R}^d$, is drawn i.i.d. from the standard d-dimensional normal distribution. We require $d \ll D$. The heavy-tailed behavior is obtained by drawing for every observation a hidden *precision scaling factor*, $\lambda_{ij} > 0$, from a gamma distribution, $\mathcal{G}(\alpha, \beta)$ parametrized by $\alpha = \beta = \frac{\nu}{2} > 0$. The parameter ν is known as the *degrees of freedom* [4, 8]. Finally, given the hidden variables, the observations are drawn i.i.d. from the multivariate normal:

$$P(\mathbf{r}_{ij} \mid \mathbf{z}_i, \lambda_{ij}) = \mathcal{N}\left(\mathbf{r}_{ij} \mid \mathbf{F}\mathbf{z}_i, (\lambda_{ij}\mathbf{W})^{-1}\right) \quad (1)$$

where $\mathbf{F}$ is the D-by-d, *factor loading matrix* and where $\mathbf{W}$ is a D-by-D positive definite *precision matrix*. The model parameters are $\nu, \mathbf{F}, \mathbf{W}$. This model is a simplification of Kenny's HT-PLDA model [4], which also had heavy-tailed speaker identity variables.

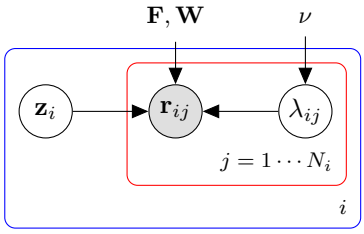

Figure 1: *Heavy-tailed PLDA*

This model does not allow closed-form scoring or training. Either the hidden scaling factors, or the hidden speaker identity variables can be integrated out in closed form, but not both. This means that we have to find approximations for both scoring and training. We make use of a new approximation, the *Gaussian likelihood approximation*, as recently published in [1]. In that paper, the approximation was used for both scoring and discriminative training. In this paper, we apply the same approximation also for generative training.

2.1. The Gaussian likelihood approximation

Both scoring and training recipes can be built around the *likelihood for the hidden speaker identity variable*, given the observation. Marginalization over the hidden variable, λ_{ij}, gives a multivariate t-distribution for the observed vector [8, 9, 1]:

$$P(\mathbf{r}_{ij} \mid \mathbf{z}_i) = \mathcal{T}(\mathbf{r}_{ij} \mid \mathbf{F}\mathbf{z}_i, \mathbf{W}, \nu) \quad (2)$$

This is a t-distribution for $\mathbf{r}_{ij}$, but to use this as likelihood for $\mathbf{z}_i$, we need to view it as function of $\mathbf{z}_i$. Provided that $D > d$ and $\mathbf{F}'\mathbf{W}\mathbf{F}$ is invertible, it is shown in [9] that this function is

10.21437/Interspeech.2018-2128

proportional to another t-distribution, with *increased* degrees of freedom, $\nu' = \nu + D - d$:

$$P(\mathbf{r}_{ij} \mid \mathbf{z}_i) \propto \mathcal{T}(\mathbf{z} \mid \hat{\mathbf{z}}_{ij}, \mathbf{B}_{ij}, \nu') \tag{3}$$

where

$$\hat{\mathbf{z}}_{ij} = \mathbf{B}_{ij}^{-1}\mathbf{a}_{ij} \qquad \mathbf{B}_{ij} = b_{ij}\mathbf{B}_0, \tag{4}$$

$$\mathbf{a}_{ij} = b_{ij}\mathbf{F}'\mathbf{W}\mathbf{r}_{ij}, \qquad b_{ij} = \frac{\nu + D - d}{\nu + \mathbf{r}_{ij}'\mathbf{G}\mathbf{r}_{ij}}, \tag{5}$$

$$\mathbf{B}_0 = \mathbf{F}'\mathbf{W}\mathbf{F}, \qquad \mathbf{G} = \mathbf{W} - \mathbf{W}\mathbf{F}\mathbf{B}_0^{-1}\mathbf{F}'\mathbf{W} \tag{6}$$

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 likelihood practically Gaussian. We therefore approximate the speaker identity likelihood as:

$$P(\mathbf{r}_{ij} \mid \mathbf{z}) \approx \exp\big[\mathbf{a}_{ij}'\mathbf{z} - \frac{1}{2}\mathbf{z}'\mathbf{B}_{ij}\mathbf{z}\big] \propto \mathcal{N}(\mathbf{z} \mid \hat{\mathbf{z}}_{ij}, \mathbf{B}_{ij}^{-1}) \tag{7}$$

Notice that for the heavy-tailed case, with small ν, the likelihood precisions are variable, while in the Gaussian limit, at $\nu \to \infty$, we have $b_{ij} = 1$ and constant precisions. The precision variability is driven by $\mathbf{r}_{ij}'\mathbf{G}\mathbf{r}_{ij}$, where $\mathbf{G}$ is a projection operator onto the orthogonal complement of the speaker subspace[1]—with the inner product defined by the positive definite precision matrix [10].

2.2. Scoring

Since the precisions of all likelihoods extracted by this model differ only by a scale factor, b_{ij}, they can be jointly diagonalized, to give fast scoring of speaker verification trials as explained in more detail in [1, 9].

2.3. Mean-field VB training

The hidden variables associated with speaker i are $\mathbf{z}_i$ and $\boldsymbol{\lambda}_i = \{\lambda_{ij}\}_{j=1}^{N_i}$. These hidden variables are *dependent* in the true joint posterior, $P(\mathbf{z}_i, \boldsymbol{\lambda}_i \mid \mathcal{R}_i)$ and this posterior does not have a closed form. We follow [4] and use mean-field variational Bayes (VB) [8] that makes use of an approximate, factorized posterior of the form:

$$Q_i(\boldsymbol{\lambda}, \mathbf{z}) = Q_i(\boldsymbol{\lambda})Q_i(\mathbf{z}) \approx P(\mathbf{z}, \boldsymbol{\lambda} \mid \mathcal{R}_i) \tag{8}$$

Given this factorization, the *VB lower bound* is formed as:

$$\mathcal{L} = \sum_i \left\langle \log \frac{P(\mathcal{R}_i, \mathbf{z}, \boldsymbol{\lambda} \mid \theta)}{Q_i(\mathbf{z})Q_i(\boldsymbol{\lambda})} \right\rangle_{Q_i(\mathbf{z})Q_i(\boldsymbol{\lambda})} \tag{9}$$

where the model parameters are $\theta = (\mathbf{F}, \mathbf{W}, \nu)$; and where $\mathcal{L} \leq P(\mathcal{R} \mid \theta)$, the true marginal likelihood. Training is done by maximizing $\mathcal{L}$ w.r.t. both θ and the Q-factors, to give an approximation to $\operatorname{argmax}_\theta P(\mathcal{R} \mid \theta)$.

In the traditional mean-field VB recipe, $\mathcal{L}$ is optimized iteratively, doing partial maximizations w.r.t. θ, the $Q_i(\boldsymbol{\lambda})$ and the $Q_i(\mathbf{z})$ in round-robin fashion. The Q-factor optimizations are variational, rather than parametric. The *forms* of the optimal Q-factors can be derived in closed form: multivariate Gaussian for $Q_i(\mathbf{z})$; and product of independent gamma distributions for $Q_i(\boldsymbol{\lambda})$. But for every i, the parameters of these distributions have to be iteratively computed. This makes the traditional recipe slow. To get a fast alternative, we choose a closed form for the one Q-factor:

$$Q_i(\boldsymbol{\lambda}_i) = \prod_{j=1}^{N_i} \mathcal{G}\big(\lambda_{ij} \mid \frac{\nu + D - d}{2}, \frac{\nu + \mathbf{r}_{ij}'\mathbf{G}\mathbf{r}_{ij}}{2}\big) \tag{10}$$

This is *different* from the optimal mean-field factor, but this choice is made such that the expected values, $\langle \lambda_{ij} \rangle$ equal the b_{ij} in (5). Given (10), we can now apply the standard mean-field solution [8]:

$$\log Q_i(\mathbf{z}) = \big\langle \log P(\mathcal{R}_i, \mathbf{z} \mid \boldsymbol{\lambda}) \big\rangle_{Q_i(\boldsymbol{\lambda})} + \text{const} \tag{11}$$

to also find the other Q-factor in closed form:

$$Q_i(\mathbf{z}) \propto P(\mathbf{z}) \prod_{j=1}^{N_i} \mathcal{N}(\mathbf{z} \mid \hat{\mathbf{z}}_{ij}, \mathbf{B}_{ij}^{-1}) \propto \mathcal{N}(\mathbf{z} \mid \bar{\mathbf{z}}_i, \bar{\mathbf{B}}_i^{-1}) \tag{12}$$

where $P(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$ and

$$\bar{\mathbf{z}}_i = \bar{\mathbf{B}}_i^{-1} \sum_{j=1}^{N_i} \mathbf{a}_{ij}, \qquad \bar{\mathbf{B}}_i = \mathbf{I} + \sum_{j=1}^{N_i} \mathbf{B}_{ij} \tag{13}$$

This solution agrees with the Gaussian likelihood approximation in the following sense. Notice that the true posterior for $\mathbf{z}_i$ can be expressed as:

$$P(\mathbf{z} \mid \mathcal{R}_i) \propto P(\mathbf{z}) \prod_j P(\mathbf{r}_{ij} \mid \mathbf{z}) \tag{14}$$

If we replace the above t-distribution likelihoods, $P(\mathbf{r}_{ij} \mid \mathbf{z})$, with the Gaussian approximations in (7), then we also arrive at (12).

Our learning algorithm proceeds as follows. Fix ν to some chosen value and randomly initialize $\mathbf{F}, \mathbf{W}$. Then iterate:

E-step: Assign the $Q_i(\boldsymbol{\lambda})$ and $Q_i(\mathbf{z})$ using (10) and (12).

M-step: Maximize (9) w.r.t. $\mathbf{F}, \mathbf{W}$.

All of the above steps have closed forms and the resulting optimization algorithm proceeds very similarly to the usual EM-algorithm for training Gaussian PLDA [11]. The only difference is that the zero, first and second-order stats in the new algorithm are weighted by the scaling factors, b_{ij}. As in [11], we augment the M-step with minimum divergence [12] on the hidden variable prior $P(\mathbf{z})$. We also do minimum divergence on the hidden scale factors, as explained in *Example 2.6: Multivariate t-distribution with known degrees of freedom*, in [13]. The minimum divergence augmentations lead to much faster convergence and a well-calibrated end-result. Again, the posterior precisions, $\bar{\mathbf{B}}_i$, are mutually diagonalizable, requiring but a single eigenanalysis of $\mathbf{B}_0$ per iteration.

An open-source implementation of the training and scoring algorithms for this model is available at `github.com/bsxfan/meta-embeddings/tree/master/code/Niko/matlab/clean/VB4HTPLDA`.

3. Experiments

3.1. HT-PLDA for i-vectors

3.1.1. Experimental setup

We continue our experiments with HT-PLDA modeling of i-vectors started in [1]. Here, we keep the same experimental setup except for a few minor differences mentioned below.

[1] $\mathbf{G}\mathbf{F} = \mathbf{0}$

As before, spectral features are 60-dimensional MFCCs with short-term mean and variance normalization applied over a 3 second sliding window. The UBM is gender independent and has 2048 diagonal components. The i-vectors are of dimension $D = 600$. We applied global mean normalization to these i-vectors (because our HT-PLDA model does not have a mean parameter). The G-PLDA backend was applied to i-vectors both with and without length normalization (LN). All HT-PLDA backends were applied to i-vectors without LN.

UBM, i-vector extractor and both Gaussian and heavy-tailed PLDAs are trained on the PRISM dataset [14], containing Fisher parts 1 and 2, Switchboard 2, 3 and Switchboard cellphone phases. Also, NIST SRE 2004–2008 (also known as 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 the full set to train the i-vector extractor. When training PLDA models, we filter out all speakers having less than 6 utterances, resulting in just 3429 speakers and 73306 training utterances.

We evaluate performance on the female part of NIST SRE'10, condition 5, which consists of English telephone data [15]. Additionally, we report the results on the NIST SRE'16 evaluation set (both males and females). We report the results on the whole evaluation set as well as on 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 of interest in the NIST SRE'16 [16] ($C_{\min}^{\text{Prm}}$).

3.1.2. Experiments and results

Our i-vector experiments, comparing traditional G-PLDA with discriminatively and generatively trained HT-PLDA are shown in table 1. The speaker subspace dimensionality is $d = 200$. The first two lines show the G-PLDA baseline, with and without length normalization. As expected, length normalization helps to shape the distribution of i-vectors to better fit Gaussian assumptions made by G-PLDA. Consequently, the performance of G-PLDA is significantly worse when no length normalization is applied.

The third and fourth lines repeat our experiments from [1]. Line 3 shows HT-PLDA with $\nu = 2$ and with $\mathbf{F}, \mathbf{W}$ simply initialized from G-PLDA.[2] Line 4 shows the same model after additional discriminative training with binary cross-entropy (BXE). In both cases no length normalization was applied. In the absence of LN, introducing the heavy-tailed mechanism at test time (line 3) is able to significantly improve the performance (compared to line 2), even without further training. Discriminatively trained HT-PLDA, without length norm (line 4) does best and surpasses the performance of G-PLDA with length normalization (line 1).

The last two lines of table 1 present results for generative VB training of HT-PLDA as described in section 2.3. In line 5, training was done with $\nu \to \infty$ and testing with $\nu = 2$. In line 6, both training and testing had $\nu = 2$. Ideally, lines 5 and 3 should be identical, but due to details of the respective EM and VB algorithms, small differences remain, possibly because the algorithms were stopped before fully converged. Although the VB trained HT-PLDA variants (lines 5, 6) do better than G-PLDA baseline without LN (line 2), it does not manage to improve the performance of G-PLDA with LN (line 1), nor of the discriminatively trained HT-PLDA (line 4).

[2] HT-PLDA with $\nu \to \infty$ is equivalent to G-PLDA.

3.2. HT-PLDA for x-vectors

3.2.1. x-vector extractor

The x-vector system is a modified version of the DNN in [6]. The features are 23 dimensional MFCCs with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds. An energy SAD is used to filter out nonspeech frames. The first few layers of the x-vector extractor operate on sequences of frames. They are a hierarchy of convolutional layers (only convolving in time) that provide a long temporal context (23 frames, 11 to each side of the center frame) with reduced complexity. Their outputs are processed by fully connected layers and followed by a statistics pooling layer that aggregates across time by computing the mean and standard deviation. This process aggregates information so that subsequent layers operate on the entire segment. The mean and standard deviation are concatenated together and propagated through segment-level layers and finally the softmax output layer. The nonlinearities are all rectified linear units (ReLUs).

The DNN is trained to classify the N speakers in the training data. A training example consists of a chunk of speech features (about 3 seconds average), and the corresponding speaker label. After training, x-vectors (512 dimension) are extracted from the affine layer immediately after the pooling layer.

The software framework has been made available in the Kaldi toolkit. An example recipe is in the main branch of Kaldi at `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2` and a pretrained x-vector system can be downloaded from `http://kaldi-asr.org/models.html`.

3.2.2. Experimental setup

The DNN training data consists of both telephone and microphone speech (mostly English). All wideband audio is downsampled to 8kHz. We pooled data from Switchboard, Fisher, Mixer (SRE 2004-2010), and VoxCeleb[3] [17] datasets yielding approximately 175K recordings from 15K speakers. Additionally, the recordings were augmented (using noise, reverb, and music) to produce 450K examples. From this augmented set, 15K chunks of 2 to 4 seconds were extracted for each speaker to form minibatches (64 chunks). We sampled equally for each speaker (i.e., balanced the training data per speaker) and trained for 3 epochs.

The G-PLDA and HT-PLDA classifiers are trained on a subset of the augmented data (we removed Switchboard and Fisher data) comprising 7K speakers and 230K recordings. For all experiments, we use a speaker subspace of dimension $d = 150$. To explore the effects of LN on x-vectors we present results with and without it. More precisely, although LN comprises multiple steps (center, whitening, and projection onto unit-sphere) we use the notation "no LN" to refer to the lack of projection. We always center and whiten the data. Finally, the scores are normalized using adaptive symmetric score normalization (as-norm) [18].

We report results on SITW core-core condition [19] and the Cantonese subset of NIST SRE'16 [16] to characterize the system behavior under microphone and telephone recording conditions. Each of these sets provides development data that we use for centering the evaluation data and computing as-norm. The

[3] We removed the 60 speakers that overlap with SITW since we evaluate on it.

Table 1: *Comparison of error-rates on SRE'10 and '16 of Gaussian PLDA with length normalization and without it, versus discriminatively and generatively trained heavy-tailed PLDA (without length normalization) using i-vectors. The performance metrics are* $C_{\min}^{\text{Prm}}$*,* *and* EER(%)

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
G-PLDA, LN	0.26	2.5	0.97	16.5	**0.68**	9.7	0.99	21.0
G-PLDA, no LN	0.33	4.0	0.97	17.8	0.69	11.5	0.98	21.3
HT-PLDA $\nu = 2$, initialized from G-PLDA	0.30	2.9	0.96	16.7	0.68	10.0	0.98	21.1
HT-PLDA $\nu = 2$, trained with BXE	**0.21**	**2.1**	**0.90**	**15.1**	0.74	**9.3**	**0.97**	**20.2**
HT-PLDA, train $\nu = \infty$, test $\nu = 2$	0.30	2.7	0.97	16.7	0.68	10.0	0.98	21.2
HT-PLDA, train $\nu = 2$, test $\nu = 2$	0.31	3.2	0.97	16.9	0.69	10.4	0.99	21.3

Table 2: *G-PLDA vs generatively trained HT-PLDA on eval part of SITW core-core using x-vectors.*

System	$\text{minDCF}_{0.01}$	EER
G-PLDA, LN	0.34	3.3
G-PLDA, no LN	0.34	3.4
HT-PLDA, $\nu = 2$, LN	0.34	3.4
HT-PLDA, $\nu = 2$, no LN	**0.33**	**2.7**

Table 3: *G-PLDA vs generatively trained HT-PLDA (with and without adaptation) on SRE'16 Cantonese using x-vectors. The performance metrics are balanced* $C_{\min}^{\text{Prm}}$*, as computed by NIST scoring tool,* $\text{minDCF}_{0.01}$ *and* EER(%)

System	$C_{\min}^{\text{Prm}}$(bal.)	$\text{minDCF}_{0.01}$	EER
G-PLDA, LN	0.30	0.31	4.5
G-PLDA, no LN	0.33	0.32	4.7
HT-PLDA, $\nu = 2$, LN	0.31	0.31	4.5
HT-PLDA, $\nu = 2$, no LN	**0.30**	**0.30**	**3.8**
+ unsupervised adaptation			
G-PLDA, LN	0.27	0.27	3.9
G-PLDA, no LN	0.29	0.28	4.3
HT-PLDA, $\nu = 2$, LN	0.27	0.28	4.2
HT-PLDA, $\nu = 2$, no LN	**0.25**	**0.26**	**3.2**

PLDA training set is always centered to its own mean and used to estimate the whitening transform. Note that this transformation does not have any impact if no projection is applied to the x-vectors. Additionally, for the SRE'16 set, we also show results applying unsupervised domain adaptation [20] of the PLDA parameters.

3.2.3. Results

The x-vector results are presented in tables 2 and 3. To the best of our knowledge, these are the best numbers published on both tasks. Moreover, the HT-PLDA classifier with no LN outperforms G-PLDA (even with domain adaptation for SRE'16). It is interesting to note that LN is detrimental to the HT-PLDA performance. Recall that the precision scaling factors b_{ij} in (5) are determined by $\mathbf{r}'_{ij}\mathbf{G}\mathbf{r}_{ij}$, the energy of the x-vectors in the complement of the speaker subspace. This results in scaling factors that get smaller as the energy of the x-vectors outside of the speaker subspace grows (which is consistent with the phenomenon that our model is trying to capture). Therefore, projecting the x-vector onto the unit sphere interferes with this process and the results indicate that it is detrimental. The G-PLDA classifier seems suboptimal for these tasks, but still benefits from LN. This is more noticeable for the SRE'16 results than for SITW where LN does not seem to have much effect. This is an indication that x-vectors behave differently than i-vectors in this regard and requires further investigation. Finally, unsupervised domain adaptation using parameter interpolation works quite well for both G-PLDA and the HT-PLDA model.

4. Discussion

In this paper and in our previous work [1], we revisit heavy-tailed PLDA and re-engineer it to provide a computationally attractive alternative to the existing state of the art given by Gaussian PLDA with length normalization. Our experiments show benefits to HT-PLDA on both i-vectors and x-vectors on three different evaluation sets. In the case of i-vectors, discriminative training worked better than generative training. In the case of x-vectors, only generative training was tried to date, and this gave record performance on SRE'16 and SITW.

In future work, we plan to try discriminative training for the HT-PLDA backend also on x-vectors. After that, we want to backpropagate the discriminative training, *through* this backend and also into the x-vector extractor. The idea is that the variable precisions of HT-PLDA should serve as a vehicle for uncertainty propagation from the input MFCCs to the output scores, as more fully motivated in [1, 9].

5. 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. The work was supported by 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".

6. References

[1] N. Brümmer, A. Silnova, L. Burget, and T. Stafylakis, "Gaussian meta-embeddings for efficient scoring of a heavy-tailed plda model," in *Odyssey: The Speaker and Language Recognition Workshop*, Les Sables d'Olonne, 2018. [Online]. Available: arxiv.org/abs/1802.09777

[2] 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.

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

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

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

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

[7] 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.

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

[9] N. Brümmer, L. Burget, P. Garcia, O. Plchot, J. Rohdin, D. Garcia-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.

[10] R. T. Behrens and L. L. Scharf, "Signal processing applications of oblique projection operators," *IEEE Transactions on Signal Processing*, vol. 42, no. 6, pp. 1413–1424, Jun 1994.

[11] N. Brümmer, "EM for Simple PLDA," Agnitio Research, South Africa, Tech. Rep., November 2010. [Online]. Available: sites.google.com/site/nikobrummer/EMforSPLDA.pdf

[12] ——, "A minimum divergence recipe for VBEM," Agnitio Research, South Africa, Tech. Rep., October 2010. [Online]. Available: sites.google.com/site/nikobrummer/VBEMandMINDIV.pdf

[13] G. J. McClachlan and T. Krishnan, *The EM Algorithm and Extensions*, 2nd ed. John Wiley & Sons, 2008.

[14] 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.

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

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

[17] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Interspeech*, 2017.

[18] D. Sturim and D. Reynolds, "Speaker adaptive cohort selection for tnorm in text-independent speaker verification," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 1. IEEE, 2005, pp. I–741.

[19] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The 2016 speakers in the wild speaker recognition evaluation." in *Interspeech*, 2016, pp. 823–827.

[20] D. Garcia-Romero, A. McCree, S. Shum, N. Brümmer, and C. Vaquero, "Unsupervised domain adaptation for i-vector speaker recognition," in *Odyssey: The Speaker and Language Recognition Workshop*, 2014.