Probabilistic embeddings for speaker diarization

Anna Silnova[1], Niko Brümmer[2], Johan Rohdin[1,2], Themos Stafylakis[2], Lukáš Burget[1]

[1]Brno University of Technology, FIT, IT4I CoE, Czechia
[2]Omilia - Conversational Intelligence, Athens, Greece
{isilnova,rohdin,burget}@fit.vutbr.cz, {nbrummer,tstafylakis}@omilia.com

Abstract

Speaker embeddings (x-vectors) extracted from very short segments of speech have recently been shown to give competitive performance in speaker diarization. We generalize this recipe by extracting from each speech segment, in parallel with the x-vector, also a diagonal precision matrix, thus providing a path for the propagation of information about the quality of the speech segment into a PLDA scoring backend. These precisions quantify the uncertainty about what the values of the embeddings might have been if they had been extracted from high quality speech segments. The proposed *probabilistic embeddings* (x-vectors with precisions) are interfaced with the PLDA model by treating the x-vectors as hidden variables and marginalizing them out. We apply the proposed probabilistic embeddings as input to an agglomerative hierarchical clustering (AHC) algorithm to do diarization in the DIHARD'19 evaluation set. We compute the full PLDA likelihood 'by the book' for each clustering hypothesis that is considered by AHC. We do joint discriminative training of the PLDA parameters and of the probabilistic x-vector extractor. We demonstrate accuracy gains relative to a baseline AHC algorithm, applied to traditional x-vectors (without uncertainty), and which uses averaging of binary log-likelihood-ratios, rather than by-the-book scoring.

1. Introduction

We are interested in doing speaker diarization by extracting embeddings from very short (1.5s or shorter) speech segments, followed by clustering of the embeddings w.r.t. speaker. To process a speech recording for diarization, we use the following cascade: (i) a deep neural net (DNN) to extract embeddings from short consecutive segments of speech; (ii) a probabilistic discriminant analysis (PLDA) backend that processes the embeddings to compute likelihoods for hypothesized speaker clusters; (iii) agglomerative hierarchical clustering (AHC) to greedily find an approximately optimal clustering. This is one of the standard ways of doing diarization, but we propose modifications to all three stages.

Intuitively, for speaker diarization purposes, the quality of speech segments could depend on factors like speech duration, reverberation, microphone placement, signal-to-noise ratio and contamination by overlapped speech. We do not propose here to explicitly design extractors for such quality factors, but we rely instead on the ability of DNNs to learn to extract relevant information from the input.

By treating the embeddings as hidden variables, we derive a mechanism to propagate the quality information, that is supplied by the embedding extractor, into the scoring and clustering backend. This gives an augmented embedding extractor that computes from each speech segment not only an x-vector, but also a precision matrix that quantifies the uncertainty about

what the value of the x-vector might have been if it had been extracted from a high quality speech segment. We show how to modify both PLDA and AHC to make use of this extra information.

We discriminatively train the new embedding extractor jointly with the PLDA parameters, using a novel multiclass cross-entropy criterion formed by the average log-posterior probabilities for the correct clustering of n-tuples of embeddings. In this paper we use $n = 8$.

The setup for this work is derived from our winning submission to DIHARD'19 challenge, where x-vectors were clustered by Bayesian HMM (BHMM) [1]. Although in this paper we use the simpler AHC clustering, our proposed mechanisms of quality propagation could be applied to BHMM in future.

Section 2 reviews prior work. Section 3 develops the theory to marginalize over the hidden embeddings and to score the modified PLDA backend. This sets the stage for discriminative training and AHC clustering in sections 4 and 5. Finally, experiments and results in section 6 demonstrate the theory.[1]

2. Prior work

Space constraint precludes a review of the extensive literature on speaker diarization. The focus and novelty of this paper is the probabilistic embedding mechanism, which we place into context relative to similar prior work.

In i-vector speaker recognition, the quality effects of duration and phonetic variability were propagated into the PLDA backend in [2, 3]. We cannot build on these works, since they relied on the i-vector posterior uncertainty as defined by the *generative* i-vector extraction model, while the x-vector extractor that we use here is *discriminative*. Our method allows propagation of more general quality factors.

In our previous work [4], we propagated scalar-valued uncertainties extracted from i-vector/x-vector magnitudes into a heavy-tailed PLDA backend, while the embedding extractor remained as-is. Our new work is more general in the following aspects: (i) The embedding extractor is modified and trained to extract (ii) vector-valued uncertainty; (iii) The hidden variable, around which the propagation hinges, is now the embedding, rather than the speaker identity variable as in [4].

In [5] *posterior probability distributions* for embeddings given images are proposed. Similar to our approach, they treat the embeddings as hidden variables. In contrast, we work with *likelihood distributions*, the advantages of which are discussed later.

In [6] an embedding extractor was discriminatively trained for diarization. They did not use uncertainty propagation and their training criterion (pseudolikelihood) was different from
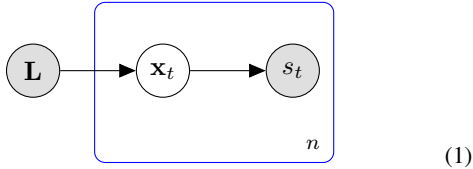
---

[1]https://github.com/bsxfan/probabilistic_embeddings

ours. It should be noted that both their pseudolikelihood [7] and our cross-entropy are proper scoring rules [8] that encourage good calibration of the trained models.

3. Theory

In this section we present a novel way to derive a discriminatively trainable embedding extractor from a generative model. The generative approach facilitates a principled interface between the extracted embeddings and the PLDA backend. Although the whole model is generative, we show how to effectively ignore parts of the model so that it can be discriminatively trained. After training, the same model can be used at runtime for diarization. We make our modelling assumptions subject to tractability and computational efficiency constraints.

Let $\mathbf{S} = \{s_t\}_{t=1}^n$ denote an n-tuple of speech segments, for which we consider the following generative model. First, a speaker clustering hypothesis [9], denoted $\mathbf{L}$, is generated from some prior distribution, $P(\mathbf{L})$. Then $\mathbf{L}$ generates an n-tuple of *hidden embeddings*, $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^n$, where $\mathbf{x}_t \in \mathbb{R}^D$. Finally each speech segment, s_t, is generated from the associated $\mathbf{x}_t$:

$$(1)$$

The s_t are always observed, $\mathbf{L}$ is observed at training time, but not at runtime, while the $\mathbf{x}_t$ are always hidden. Notice that according to this model, the $\mathbf{x}_t$ would be the *ideal* features for inference of $\mathbf{L}$ from $\mathbf{S}$, because of the conditional independence:

$$P(\mathbf{L} \mid \mathbf{X}, \mathbf{S}) = P(\mathbf{L} \mid \mathbf{X}) \qquad (2)$$

Unfortunately, these ideal features cannot be directly observed or computed and we have to resort to marginalization to compute the *clustering posterior*:

$$P(\mathbf{L} \mid \mathbf{S}) = \int P(\mathbf{L}, \mathbf{X} \mid \mathbf{S}) \, d\mathbf{X} \qquad (3)$$

The rest of this section is devoted to constructing a model, with associated tractable algorithms, that enables computation of (3).

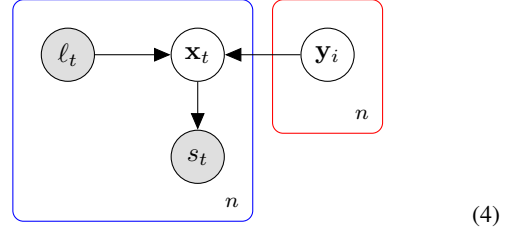### 3.1. The partition prior

The Bell number, B_n, gives the number of ways that n items can be partitioned (clustered). For octets, we have $B_8 = 4140$. The support of the partition prior, $P(\mathbf{L})$, can include up to B_n discrete possibilities, although we might sometimes consider priors with fewer possibilities—e.g. if we know there are fewer than n speakers.

In this paper, we use a Chinese restaurant process (CRP) to define $P(\mathbf{L})$. The CRP assumes that the sequence of speaker labels are exchangeable and that time plays no role. More details of our CRP prior are given later. In future work, to model dependency between consecutive segments, $P(\mathbf{L})$ could be defined by a Markov model [1], or perhaps by a distance-dependent CRP [10].

3.2. The PLDA model

We introduce the PLDA backend via a more detailed view of model (1), which now includes also n *speaker identity variables*, $\mathbf{y}_i \in \mathbb{R}^d$. For simplicity in defining this model, we

let the maximum number of speakers be equal to the number of speech segments, n. The clustering hypothesis is represented here as set of *speaker labels*, $\mathbf{L} = \{\ell_t\}_{t=1}^n$, where each $\ell_t \in \{1, \ldots, n\}$.

$$(4)$$

Only a subset of the speakers might produce observations: There is an arc from every $\mathbf{y}_i$ to every $\mathbf{x}_t$, but when $\ell_t = k$, then only $\mathbf{y}_k$ generates observation $\mathbf{x}_t$.

The label set, $\mathbf{L} = \{\ell_t\}_{t=1}^n$, requires some further explanation. In the case of octets for example, there are $B_8 = 4140$ possible ways to cluster 8 elements. But the label set, if it were unconstrained, would have 8^8 (almost 17 million) possibilities. We can however arrange a deterministic, one-to-one relationship between $\mathbf{L}$ and the possible clusterings by constraining $\mathbf{L}$ to be a *restricted growth string* [11], subject to the constraints:

$$\ell_1 = 1 \qquad \text{and} \qquad \ell_{t+1} \leq 1 + \max(\ell_1, \ldots, \ell_t) \quad (5)$$

The $\mathbf{y}_i$ are generated IID from the standard d-dimensional multivariate Gaussian, while the $\mathbf{x}_t$ are generated from the D-dimensional conditional Gaussian:

$$\mathbf{x}_t \mid (\mathbf{y}_k, \ell_t = k) \sim \mathcal{N}(\mathbf{V}\mathbf{y}_k, \mathbf{W}^{-1}) \qquad (6)$$

where $\mathbf{V}$ is the D-by-d speaker factor loading matrix and $\mathbf{W}$ is the D-by-D within-speaker precision matrix. If $D > d$, this model is usually referred to as simple PLDA (SPLDA), while if $D = d$ and $\mathbf{V}$ is of full rank, the model is referred to as the *two covariance model* [9], where $\mathbf{V}\mathbf{V}'$ is the between-speaker covariance.

3.3. Marginalization

The marginalization of each $\mathbf{x}_t$ can be done independently of its siblings, by computing the *speaker identity likelihood*, given segment s_t:

$$\begin{aligned} P(s_t \mid \mathbf{y}) &= \int P(s_t, \mathbf{x}_t \mid \mathbf{y}) \, d\mathbf{x}_t \\ &= \int P(\mathbf{x}_t \mid \mathbf{y}) P(s_t \mid \mathbf{x}_t) \, d\mathbf{x}_t \end{aligned} \qquad (7)$$

where $\mathbf{y}$ is the identity variable of the speaker of segment t. The first factor in the final integrand is Gaussian, given by (6). If we can force the second factor, $P(s_t \mid \mathbf{x}_t)$, the *likelihood* for $\mathbf{x}_t$ given s_t, to be Gaussian as a function of $\mathbf{x}_t$, then the integral can be computed in closed form. We shall do exactly that by imposing a constraint on our model. But first, we need to consider (carefully) the following factorization of the likelihood, which can be done without loss of generality:

$$\begin{aligned} P(s_t \mid \mathbf{x}_t) &= \frac{P(s_t)}{g(s_t)} \times g(s_t) \frac{P(\mathbf{x}_t \mid s_t)}{P(\mathbf{x}_t)} \\ &= h(s_t) \ \times f(s_t, \mathbf{x}_t) \end{aligned} \qquad (8)$$

where we have introduced the real-valued functions f, g, h, such that g, h are strictly positive and f is non-negative. The

function $f(s_t, \mathbf{x}_t)$ is a formal representation of our embedding extractor, a trainable DNN, that implements the mapping: $s_t \mapsto f(s_t, \cdot)$, where the output (a likelihood distribution) is the *probabilistic embedding* that is extracted from s_t. Both $P(s_t \mid \mathbf{x}_t)$ and $f(s_t, \mathbf{x}_t)$ are valid forms for the likelihood for $\mathbf{x}_t$ given s_t. The arbitrary functions g, h serve to express in full generality the unnormalized nature of likelihood distributions.

The difference between the probabilistic embeddings of [5] and ours is shown by:

$$f(s_t, \mathbf{x}_t) = g(s_t)\frac{P(\mathbf{x}_t \mid s_t)}{P(\mathbf{x}_t)} \tag{9}$$

We use the likelihood, $f(s_t, \mathbf{x}_t)$, while [5] uses the posterior, $P(\mathbf{x}_t \mid s_t)$. The likelihood is unencumbered by the necessarily complex embedding prior, $P(\mathbf{x}_t)$, which in our case is defined by the PLDA backend. If your embeddings are any good at separating classes, then $P(\mathbf{x}_t)$ needs to reflect this and will take the form of a mixture with well-separated components. This is also true of the posterior, $P(\mathbf{x}_t \mid s_t)$, and indeed in [5] they are forced to work with complex probabilistic embeddings in mixture form.[2] In the likelihood form, the complex prior is effectively 'divided out' and we can work with simpler embedding distributions. We achieve more freedom in the form of the embedding by this decoupling from the backend.

As will be shown below, $h(s_t)$ cancels in our scoring formulas and therefore plays no role at runtime or during discriminative training, affording us the luxury of tractable calculations involving only f. The function, h carries all the complexities of the distribution of the raw input speech and by side-stepping it, we enjoy the discriminative privilege of not having to model the input. Nevertheless, discriminative training of a parametric form of $f(s_t, \mathbf{x}_t)$ (as we do) has some implications for (8) which are discussed further in appendix A.

We can now introduce our modelling constraint. Since $\mathbf{x}_t$ is hidden, we have considerable freedom in choosing its nature (discreet or continuous), its dimensionality and its relationship to s_t. We let $f(s, \mathbf{x})$ be Gaussian in $\mathbf{x}$, with mean and precision matrix that are arbitrarily complex functions of s. We represent the Gaussian, $f(s_t, \mathbf{x}_t)$, as:

$$f(s_t, \mathbf{x}) = \exp\left[-\frac{1}{2}\mathbf{x}'\mathbf{B}_t\mathbf{x} + \mathbf{x}'\mathbf{B}_t\hat{\mathbf{x}}_t\right] \tag{10}$$

where $\hat{\mathbf{x}}_t \in \mathbb{R}^D$ and $\mathbf{B}_t$ (positive semi-definite precision matrix) are functions of s_t. In summary, our proposed, trainable DNN embedding extractor does: $s_t \mapsto \hat{\mathbf{x}}_t, \mathbf{B}_t$. This is very similar to the encoder of a variational autoencoder [12]. Note however that the Gaussian (10) is unnormalized and in general not even normalizable, unless $\mathbf{B}_t$ is positive definite (invertible). Our recipe works in general for non-invertible $\mathbf{B}_t$, as long as the PLDA parameter, $\mathbf{W}$, is invertible.

After some algebra to solve the Gaussian integral, plugging (6), (8) and (10) into (7) gives:

$$P(s_t \mid \mathbf{y}) \propto \exp\left[-\frac{1}{2}\mathbf{y}'\mathbf{V}'(\mathbf{W} - \mathbf{W}(\mathbf{W} + \mathbf{B}_t)^{-1}\mathbf{W})\mathbf{V}\mathbf{y}\right.$$
$$\left. + \mathbf{y}'\mathbf{V}'\mathbf{W}(\mathbf{W} + \mathbf{B}_t)^{-1}\mathbf{B}_t\hat{\mathbf{x}}_t\right] \tag{11}$$

where we have omitted factors that are independent of $\mathbf{y}$, including $h(s_t)$.

---

[2]In the i-vector–PLDA cascade, the simple, single Gaussian i-vector prior is *inconsistent* with a view of the whole cascade as a single generative model. In such a model, the i-vector prior would be defined by the PLDA backend, which is impractically complex.

3.4. Diagonalization

One way to avoid the $\mathcal{O}(D^3)$ computational requirements of (11), is to specialize to the two-covariance model, with $D = d$ and full rank, square $\mathbf{V}$. Then, without loss of generality, we can linearly transform the hidden $\mathbf{x}_t$, so that $\mathbf{V}\mathbf{V}'$ and $\mathbf{W}$ are mutually diagonalized. First, transform $\mathbf{x}_t$ so that $\mathbf{V}\mathbf{V}' = \mathbf{I}$. This is followed by pure rotation that diagonalizes $\mathbf{W}$, while preserving $\mathbf{V}\mathbf{V}' = \mathbf{I}$. We are now free to let $\mathbf{V} = \mathbf{I}$. Finally, we also constrain the extracted embedding precision, $\mathbf{B}_t$ to be diagonal. We can now express (11) as the product of univariate Gaussians:

$$P(s_t \mid \mathbf{y}) \propto \prod_{j=1}^{D} \exp\left[\frac{w_j b_{jt}}{w_j + b_{jt}}(\hat{x}_{jt}y_j - \frac{1}{2}y_j^2)\right] \tag{12}$$

where $w_j > 0$ and $b_{jt} \geq 0$ are diagonal elements of $\mathbf{W}$ and $\mathbf{B}_t$, while y_j and $\hat{x}_{jt}$ are components of $\mathbf{y}$ and $\hat{\mathbf{x}}_t$. If $b_{jt} > 0$, then Gaussian j has mean $\hat{x}_{jt}$ and variance $\frac{w_j + b_{jt}}{w_j b_{jt}}$. High quality components with $b_{jt} \gg w_j$, will have low uncertainty, with variance saturated at $1/w_j$. Conversely, low quality components with $b_{jt} \ll w_j$ can have arbitrarily large uncertainty. At the limit, $b_{jt} = 0$, the associated component $\hat{x}_{jt}$ will be completely ignored in any downstream processing, as will be shown below.

3.5. The clustering posterior

The clustering (partition) posterior (3) can be expressed as:

$$P(\mathbf{L} \mid \mathbf{S}) = \frac{P(\mathbf{L})\prod_{i=1}^{n} P(\mathbf{S}^{(i)} \mid \mathbf{L})}{\sum_{\tilde{\mathbf{L}}} P(\tilde{\mathbf{L}})\prod_{i=1}^{n} P(\mathbf{S}^{(i)} \mid \tilde{\mathbf{L}})} \tag{13}$$

where the summation in the denominator is over all B_n possibilities allowed by (5) and where $P(\mathbf{S}^{(i)} \mid \mathbf{L})$ is the joint distribution for all speech segments[3] attributed to speaker i by label set $\mathbf{L}$. We expand:

$$P(\mathbf{S}^{(i)} \mid \mathbf{L}) = \int P(\mathbf{y}) \prod_{t \in \mathbf{L}_i} P(s_t \mid \mathbf{y}) \, d\mathbf{y} \tag{14}$$

where $\mathbf{L}_i$ is the set of speech segment indices attributed to speaker i by $\mathbf{L}$. For $P(s_t \mid \mathbf{y})$, we can plug the RHS of (11) or (12) into (14), because the omitted factors cancel in (13). The cancelled factors include $\prod_{t=1}^{n} h(s_t)$, which occurs in the numerator and in each term of the denominator. Recalling that $P(\mathbf{y})$ is standard Gaussian, the integrand in (14) is a product of Gaussians so that the integral can be solved in closed form. For the diagonal case, we find after some algebra:

$$\log P(\mathbf{S}^{(i)} \mid \mathbf{L}) = \frac{1}{2}\sum_{j=1}^{D}\left(\frac{\bar{a}_j^2}{1+\bar{b}_j} - \log(1 + \bar{b}_j)\right) + \text{const}$$

where

$$\bar{a}_j = \sum_{t \in \mathbf{L}_i} \frac{w_j b_{jt}}{w_j + b_{jt}}\hat{x}_{jt}, \qquad \bar{b}_j = \sum_{t \in \mathbf{L}_i} \frac{w_j b_{jt}}{w_j + b_{jt}} \tag{15}$$

The constant is independent of $\mathbf{L}$ and cancels in all our scoring and training recipes. Notice again that if $b_{jt} = 0$, then $\hat{x}_{jt}$ is effectively removed from the computation. Conversely, if $b_{jt} \gg w_j$, then the weight for $\hat{x}_{jt}$ saturates at w_j. If all b_{jt}

---

[3]If no segments are attributed to speaker i, we conveniently let $P(\mathbf{S}^{(i)} \mid \mathbf{L}) = 1$.

are sufficiently large, the result is the same as when traditional (fixed) embeddings, $\hat{\mathbf{x}}_t$, are just plugged into PLDA. The normalized, weighted sum (15) may be interpreted as an attention mechanism [13], where the w_j are the queries, the b_{jt} are the keys and the $\hat{x}_{jt}$ are the values.

3.5.1. The posterior computation recipe

In summary, to compute the clustering posterior for n-tuples, we need:

- A DNN that serves as probabilistic embedding extractor that does $s_t \mapsto \{\hat{x}_{jt}, b_{jt}\}_{j=1}^D$. We assume that the final stage of the extractor for $\hat{\mathbf{x}}_t$ is a trainable linear transform, so that we may assume the extracted embeddings will have been transformed to be compatible with a diagonalized two-covariance backend.

- A diagonalized, two-covariance PLDA model that supplies the within-speaker precision parameters, $\{w_j\}_{j=1}^D$.

- A precomputed clustering prior, $P(\mathbf{L})$, that respects (5). The prior can be stored in a table of size B_n, containing log probabilities.

- A precomputed sparse n-by-$(2^n - 1)$ matrix, with $0/1$ entries, that can be used to efficiently do the summations to compute $\bar{a}_j$ and $\bar{b}_j$ in (15), for all possible subsets $\mathbf{L}_i$. From these, we can compute $\log P(\mathbf{S}^{(i)} \mid \mathbf{L})$ for every possible subset.

- Another precomputed sparse B_n-by-$(2^n - 1)$ matrix, with $0/1$ entries, that can be used to efficiently accumulate $\sum_{i=1}^n \log P(\mathbf{S}^{(i)} \mid \mathbf{L})$, for every possible value of $\mathbf{L}$. After addition of $\log P(\mathbf{L})$, a softmax of size B_n computes the final posterior (13).

4. Discriminative training

We jointly train the embedding extractor and the PLDA parameters by minimizing the following B_n-way multiclass cross-entropy criterion, computed using (13):

$$\mathcal{C} = - \sum_{(\mathbf{L},\mathbf{S}) \in \mathcal{D}} \log P(\mathbf{L} \mid \mathbf{S}) \qquad (16)$$

where $\mathcal{D}$ is a collection of supervised *trials*, each containing an n-tuple of speech segments, $\mathbf{S}$, and the associated true speaker clustering, $\mathbf{L}$. In this paper, we choose $n = 8$ and we refer to the n-tuples as *octets*. The selection of segments to compose octet trials is described in the section on experiments below. Note that $B_n = 4140$, which gives a relatively large, but still tractable number of clustering hypotheses for each octet trial.

To compute the clustering posterior (13), we require also a clustering prior, $P(\mathbf{L})$. In this paper we use a Chinese restaurant process, which will be detailed later.

5. Diarization with Agglomerative Hierarchical Clustering

At training time we can control n, the number of speech segments to cluster, so that B_n remains tractable. At runtime, where n may be a few hundred, B_n far exceeds the number of atoms in the known universe. This makes an exact search for an optimal clustering hopelessly intractable.

Agglomerative hierarchical clustering (AHC) is a greedy algorithm for finding approximately optimal clustering solutions. For speaker diarization, AHC is initialized with one cluster per segment. Each segment is represented by an x-vector (baseline), or probabilstic x-vector (proposed). At each iteration, one pair of clusters is joined, according to some local optimality condition computed using the x-vectors. When a stopping criterion is met, each of the final clusters is attributed to a different speaker.

The baseline AHC algorithm described below cannot be interpreted as a search for the maximum-likelihood (ML) clustering, because of the way that inter-cluster comparisons are computed (via binary log-likelihood-ratio averaging). In contrast, our proposed AHC algorithm *can* be considered to be a greedy search for the ML solution.

5.1. Baseline AHC

The official Kaldi diarization recipe [14] implements the following clustering algorithm. First, some PLDA model is used to calculate log-likelihood ratio verification scores as a similarity metric for each pair of x-vectors from the given test recording. The resulting pair-wise similarity matrix is the only input to the *unweighted average linkage* AHC (also known as UPGMA). At each stage of the algorithm the highest score is selected from the similarity matrix and two clusters corresponding to that score are merged. The row and column of the similarity matrix corresponding to a new cluster is computed as an average between the original scores of the two cluster components.

The similarity score threshold σ for stopping the AHC process is estimated for each recording separately using an *unsupervised linear calibration* [15]: a GMM with two univariate Gaussian components with shared variance is trained on all the scores from the similarity matrix. The two Gaussian components are assumed to be the score distributions corresponding to the same-speaker and different-speaker x-vector pairs. Therefore, σ is set as the score for which the posterior probability of both components is 0.5 (i.e. decision threshold for the same/different-speaker maximum-a-posteriori classifier).

5.2. By-the-book AHC

Our proposed AHC is a greedy maximization of the clustering log-likelihood:

$$\mathcal{L}(\mathbf{L}) = \log P(\mathbf{S} \mid \mathbf{L}) = \sum_{i \in \text{clusters of } \mathbf{L}} \log P(\mathbf{S}^{(i)} \mid \mathbf{L}) \quad (17)$$

Here $\mathbf{S}$ represents all the speech segments in the recording and $\mathbf{L}$ is a clustering hypothesis for all these segments. Each term of $\mathcal{L}(\mathbf{L})$ can be computed by (15), up to a constant that is irrelevant when comparing hypotheses. $\mathbf{L}$ is initialized to have each segment in a separate cluster. At each iteration, we do:

$$\mathbf{L} \leftarrow \underset{\mathbf{L}' \xleftarrow{\text{join 2 clusters}} \mathbf{L}}{\operatorname{argmax}} \mathcal{L}(\mathbf{L}') \qquad (18)$$

where $\mathbf{L}'$ is restricted to merging a single pair of clusters in $\mathbf{L}$. The iteration is stopped when the new hypothesis fails to give an improvement in $\mathcal{L}(\mathbf{L})$ that exceeds a preset threshold, σ. For greedy maximum likelihood, the correct threshold is $\sigma = 0$, but we make it adjustable to help compensate for the fact that maximum likelihood ignores the prior and for any other mismatches between the model and the real data.

We implement (18) using the log-likelihood-ratios,

$$\Delta = \mathcal{L}(\mathbf{L}') - \mathcal{L}(\mathbf{L}) \qquad (19)$$

where log-likelihood terms not in the to-be-merged clusters conveniently cancel. The pair with the highest Δ is joined in

each iteration, except when the best one fails to exceed σ. For to-be-joined clusters i and j, Δ is:

$$\log \frac{P(\mathbf{S}^{(i)}, \mathbf{S}^{(j)} \mid \ell'_1 = \cdots = \ell'_{N^i+N^j})}{P(\mathbf{S}^{(i)} \mid \ell_{i,1} = \cdots = \ell_{i,N^i}) P(\mathbf{S}^{(j)} \mid \ell_{j,1} = \cdots = \ell_{j,N^j})}$$

The $\ell_{i,k} \in \mathbf{L}$ are the identical speaker labels for the N^i segments of cluster i and the same applies to cluster j. The $\ell'_k \in \mathbf{L}'$ are the new labels for the joined cluster. $\mathbf{S}^{(i)}, \mathbf{S}^{(j)}$ are the sets of speech segments assigned to each cluster. Likelihoods in the numerator and denominator are computed using (15). The computation can be done reasonably fast, by precomputing $\frac{w_j b_{jt}}{w_j + b_{jt}} \hat{x}_{jt}$ and $\frac{w_j b_{jt}}{w_j + b_{jt}}$. These quantities can be stored in matrices, where each row initially corresponds to a segment and later to a cluster. When two clusters are merged, the corresponding rows of the matrices are summed, so that each matrix will have one row fewer.

6. Experiments and results

6.1. Training and evaluation data

The deep neural network (DNN) x-vector extractor was trained on VoxCeleb 1 and 2 [16] with 1.2 million speech segments from 7146 speakers plus additional 5 million segments obtained with data augmentation. Also, VoxCeleb data are used to train the baseline PLDA model.

Data from the AMI corpus [17] were used for training PLDA and probabilistic embedding extractor. We have split the data into training and cross-validation parts. Training part uses 75% of the available data. The speakers between training and cross-validation sets are not overlapping.

Finally, we perform the diarization experiments on DIHARD 2019 development and evaluation data [18, 19]. Development set, in our case, was used for hyper-parameter tuning (such as selection of the AHC stopping threshold), otherwise, no development data were used for training system parameters. Performance is evaluated in terms of *Diarization Error Rate* (DER).

6.2. Signal processing

We used the weighted prediction error (WPE) [20, 21] method to remove late reverberation from the evaluation data. We estimated a dereverberation filter on short-time Fourier transform (STFT) spectrum for every 100 seconds block of an utterance. To compute the STFT, we used 32ms windows with 8ms shift. We set the filter length and prediction delay to 30 and 3 respectively for 16kHz. The number of iterations was set to 3.

6.3. X-vector extractor

We used the x-vector extractor from BUT's submission to the 2019 DiHard challenge [1, 22, 23]. The extractor was trained with the Kaldi toolkit [24] using the SRE'16 recipe [25] with the following modifications:

- We used 40-dimensional filterbank features generated using 16kHz sampling frequency.
- The networks was trained for 6 epochs (instead of 3).
- We used 200 frames for all training segments (instead of random durations between 200 and 400 frames).
- We sample the training segments in such a way that all regions of a recording are used equally (instead selecting the segments completely at random).

- We generated around 700 Kaldi archives such that each of them contained exactly 15 training samples from each speaker (i.e. around 107K samples in each archive).
- The network has nine TDNN layers, which sees a total context of 13 frames per side, before the statistics pooling layer.
- The input to the statistics pooling layer is the concatenation of the output from the 7th and the 9th TDNN layer.

For more details, see [1, 22, 23].

6.4. Baseline PLDA

The baseline PLDA model is trained on x-vectors extracted from $3s$ speech segments from VoxCeleb 1 and 2 and utterance IDs combined with speaker IDs serve as the class labels. Before the PLDA training, the x-vectors are centered (i.e. mean normalized), whitened (i.e. normalized to have identity covariance matrix) and length-normalized [26]. The centering and whitening transformation are estimated on the joint set of DIHARD development and evaluation data.

6.5. PLDA and embedding extractor initialization

The details of the construction and initialization of our probabilistic embedding extractor and the diagonalized PLDA model are given here. Recall that the embedding extractor does:

$$s_t \mapsto \hat{\mathbf{x}}_t, \mathbf{B}_t$$

where s_t is a speech segment, $\hat{\mathbf{x}}_t \in \mathbb{R}^D$ and $\mathbf{B}_t = \{b_{jt}\}$ is a diagonal precision matrix. The diagonalized PLDA model has a single parameter, the diagonal precision matrix, $\mathbf{W} = \{w_j\}$.

In this work, we build our probabilistic embedding extractor by modifying an existing, baseline x-vector extractor. The existing extractor *includes* the final centering, whitening and length normalization as detailed in section 6.4 above. In all our experiments the x-vector extractor, including the centering, whitening and length norm, remain fixed and are not subject to retraining.

The modifications are as follows. First, we add a linear transformation after the standard (length-normalized) x-vector, so that transformed x-vector is our desired $\hat{\mathbf{x}}_t$. Second, the output of the statistics pooling layer in the x-vector extractor is passed through an additional feed-forward neural network which outputs the diagonal precisions, $\mathbf{B}_t$. The parameters of these extra components, the linear transform and the non-linear precision extractor are subject to training in our experiments. The third trainable parameter set is the diagonal PLDA within-class covariance, $\mathbf{W}$.

We use a diagonalizing transformation of the original baseline PLDA to initialize the linear x-vector transformation and $\mathbf{W}$. Here, we decrease the dimensionality of the original x-vectors from 512 to 500, where we keep the 500 dimensions with the highest speaker variability.

As an extractor for the precisions, $\mathbf{B}_t$, we use a feed-forward neural network with a single hidden layer and the *softplus* activation function. The whole net has the stucture: linear-softplus-linear-softplus. The final softplus is needed to give non-negative precisions. As an input, this network uses the output of the statistics pooling layer of the original x-vector extractor. The first-order statistics are used as they are, while second-order statistics are inverted. That is done to speed up the training since the original statistics contain standard deviation and we are interested in the precision at the output. Also, we concatenate the statistics vector with the duration of the speech

segment. The resulting dimensionality of the input is 2049. The hidden layer is 1000 dimensional and the output has a size of $D = 500$, to agree with the size of $\hat{\mathbf{x}}_t$.

The parameters of the precision extractor network are initialized randomly in such a way that at the beginning of the training the $b_{jt} \gg w_j$ condition (see Section 3.5) would be satisfied.

So, after the initialization, the model performs very similarly to the baseline PLDA with standard x-vectors.

6.6. Parameter training

Once the model is initialized, we train it using the cross-entropy objective from Section 4. Our training examples are octets of x-vectors selected randomly from the training data. All segments in an octet come from the same recording, but it is not guaranteed that they are consecutive. Selecting the segments in a random order results in more difficult training examples and leads to faster training. (We did also try using consecutive segments and it performs similarly, except that training then takes longer.) We use *stochastic gradient descent* for the training, the parameters are updated after seeing a mini-batch of 100 examples. Since we initialize the PLDA and the transformation matrix from the existing PLDA we do not want them to deviate from the initial values too quickly so that the precision extractor would benefit from this smart initialization. In order to achieve that, the learning rate for these parameters is set to a much smaller value (10^4 times smaller in our case) than the one for precision extracting network.

In our experiments, for the prior $P(\mathbf{L})$, we used the *Chinese restaurant process* (CRP) [27, 28, 29]. We assigned the parameters of the CRP (the concentration and discount) to control the distribution of the number of speakers: The variance is maximized, subject to the constraint that expected number of speakers for N segments is the same as the true number of speakers in the training set, where N is the total size of the training set. As mentioned above, the CRP gives an exchangeable distribution that is invariant to the order of the speaker labels. This prior is appropriate for our procedure of randomly selecting the segments to form the octets during training. For situations where the order matters, other priors could be considered, for example those mentioned in section 3.1.

6.7. Results

The results are summarized in table 1 in terms of diarization error-rate (DER). For each clustering strategy (rows), the table presents three sets of results (pairs of columns). The result sets differ in how the AHC stopping threshold, σ, was set. In the first two columns, $\sigma = 0$, which is the optimal maximum-likelihood threshold.

In the second set, σ was tuned on the DIHARD development set to minimize diarization error-rate. There is an important difference between the baseline and the other systems: The scores for the merging decisions in the baseline AHC are individually calibrated for each recording, using the unsupervised calibration as explained above. The globally optimized threshold, σ, was applied to these calibrated scores. While for the rest of the experiments no such calibration was done and σ was applied directly to (19).

Finally, to compensate for the fact that the segments in the evaluation data are not independent, one can scale down $\log P(s_t \mid \mathbf{y})$, the speaker identity log-likelihoods of the individual speech segments [30, 31]. For us that means scaling the statistics $\bar{a}_j$ and $\bar{b}_j$ in (15). The last two columns of the ta-

ble show the results with the tuned likelihood scalar. The AHC stopping threshold, in this case, is set to $\sigma = 0$. One can notice that optimally scaling the likelihoods has a similar effect to optimizing σ.

The first row in the table presents the baseline approach. The baseline is Kaldi-style AHC (section 5.1), with the same PLDA and x-vector extractor we use to initialize our model. Here, an important note is that principal component analysis (PCA) is applied to the x-vectors, *independently* for each recording, giving a different PCA projection for each recording. The PCA output dimension is adaptively chosen so that the resulting low-dimensional projected x-vector retain only 30% of their variability. The projected x-vectors are then length normalized again prior to the clustering. The PLDA parameters are also projected to the corresponding low-dimensional space. The resulting PLDA model is used to calculate log-likelihood ratio verification scores as the similarity metric that is applied to all pairs of x-vectors in the recording. The resulting similarity matrix is the input to the AHC. If instead, omit these extra PCA and length normalization steps (as in the rest of the experiments), the baseline results degrade to 28.63% and 28.36% DER on development and evaluation sets respectively.

The rest of table 1 shows the results for the diarization with our version of AHC (section 5.2), where we use by-the-book scoring at each stage of the clustering. First (table row 2), we applied AHC with exact scoring to the untrained model initialized from the baseline. As mentioned earlier, the model before training is practically the same as the baseline. So, the main source of performance difference is the alternative way of clustering. This model is heavily miscalibrated and using $\sigma = 0$ as a threshold results in a performance inferior to the baseline. But, by tuning either the threshold or the likelihood scaling, one can achieve an improvement over the standard AHC clustering.

Next (table row 3), we present the results obtained with a system where the part of the network that extracts precisions remains fixed, so that it keeps extracting large precisions, b_{jt}. We retrain only the PLDA (the w_j) and the linear transformation (in the extractor) that outputs $\hat{\mathbf{x}}_t$. One can see that the calibration problem in this case is not that prominent anymore. But still, there is a possibility to improve the performance by carefully optimizing the threshold or the likelihood scalar.

Finally (table row 4), we enable training of all three parameter sets: the linear transform, the precision extractor and the PLDA. Here, even without tuning of the hyper-parameters, we improve over the baseline, although threshold or likelihood scalar tuning can still give small improvements.

An interesting observation is that as we move from untrained model to the one with PLDA training to the one with all components trained, the optimal values for the threshold are approaching zero. Similarly for the likelihood scaling factor, for each next system it is closer to 1.

7. Conclusions

We have made the following contributions:

- We have presented a novel methodology, developed around the factorization (8), to derive discriminatively trainable embedding extractors from generative models.

- We have motivated and experimentally demonstrated arguments for data representations in the form of probabilistically distributed hidden embeddings. In contrast to variational autoencoders [12] and other works on probabilistic embeddings, e.g. [5], that extract hidden vari-

Table 1: Diarization error-rate [%] on DIHARD 2019 development and evaluation sets.

	$\sigma=0$		σ optimal		likelihood scaling optimal	
	dev	eval	dev	eval	dev	eval
Baseline AHC	-	-	25.12	26.33	-	-
by-the-book AHC	44.28	41.09	23.82	23.98	22.85	25.18
by-the-book AHC, PLDA trained	26.95	26.34	22.38	22.68	21.05	21.86
by-the-book AHC, PLDA + embedding extractor trained	22.95	23.38	21.61	21.84	20.95	21.79

able *posterior* distributions, we extract *likelihood* distributions.

- We have applied the above to the x-vector–PLDA cascade, showing how to adapt PLDA to score probabilistic embeddings in a computationally efficient way.

- For speaker diarization, we have introduced a new discriminative training objective, a large multiclass cross-entropy, which is a proper scoring rule [8] that encourages well-calibrated likelihood distributions [32] over all the B_n possible ways to cluster an n-tuple of speech segments according to speaker. This was experimentally demonstrated on joint discriminative training of the embedding extractor and the PLDA backend.

- A new AHC algorithm that makes full use of the PLDA model to greedily optimize the maximum likelihood clustering hypothesis.

In future, we would like to make the embedding extractor more powerful by making the precision extractor deeper and also retraining the parameters of the whole x-vector extractor. The extra capacity can be regularized by more aggressive augmentation of the input segments to provide a greater variety of durations, SNR and other quality factors. We would also like to try the uncertainty propagation into the BHMM backend.

8. Acknowledgements

9. References

[1] Federico Landini, Shuai Wang, Mireia Diez, Lukáš Burget, Pavel Matějka, Kateřina Žmolíková, Ladislav Mošner, Oldřich Plchot, Ondřej Novotný, Hossein Zeinali, and Johan Rohdin, "BUT System Description for DIHARD Speech Diarization Challenge 2019," *arXiv preprint arXiv:1910.08847*, 2019.

[2] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md. Jahangir Alam, and Pierre Dumouchel, "PLDA for speaker verification with utterances of arbitrary duration," in *ICASSP, Vancouver*, 2013.

[3] Pietro Laface Sandro Cumani, Oldřich Plchot, "On the use of i–vector posterior distributions in probabilistic linear discriminant analysis," *IEEE Trans. Audio, Speech and Language Processing*, 2014.

[4] Niko Brümmer, Anna Silnova, Lukáš Burget, and Themos Stafylakis, "Gaussian meta-embeddings for efficient scoring of a heavy-tailed PLDA model," in *Odyssey 2018: The Speaker and Language Recognition Workshop*, 2018.

[5] Seong Joon Oh, Kevin Murphy, Jiyan Pan, Joseph Roth, Florian Schroff, and Andrew Gallagher, "Modeling uncertainty with hedged instance embedding," 2018.

[6] Alan McCree, Gregory Sell, and Daniel Garcia-Romero, "Speaker diarization using leave-one-out Gaussian PLDA clustering of DNN embeddings," in *INTERSPEECH, Graz*, 2019.

[7] A. Philip Dawid, Steffen Lauritzen, and Matthew Parry, "Proper local scoring rules on discrete sample spaces," *The Annals of Statistics*, 2012.

[8] Tilmann GNEITING and Adrian E. RAFTERY, "Strictly proper scoring rules, prediction, and estimation," *JASA*, vol. 102, no. 477, 2007.

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

[10] David M. Blei and Peter I. Frazier, "Distance dependent Chinese restaurant processes," *JMLR*, vol. 12, 2011.

[11] Michael Orlov, "Efficient generation of set partitions," Tech. Rep., Computer Science Department of Ben-Gurion University in Israel, March 2002.

[12] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[14] Gregory Sell, David Snyder, Alan McCree, Daniel Garcia-Romero, Jesús Villalba, Matthew Maciejewski, Vimal Manohar, Najim Dehak, Daniel Povey, Shinji Watanabe, et al., "Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge.," in *Proceedings of Interspeech 2018*, 2018, pp. 2808–2812.

[15] Niko Brümmer and Daniel Garcia-Romero, "Generative modelling for unsupervised score calibration," in *ICASSP, Florence*, 2014.

[16] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[17] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Iain McCowan, Wilfried Post, Dennis Reidsma, and

Pierre Wellner, "The ami meeting corpus: A pre-announcement," in *Proceedings of the Second International Conference on Machine Learning for Multimodal Interaction*, Berlin, Heidelberg, 2006, MLMI'05, pp. 28–39, Springer-Verlag.

[18] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "First DIHARD Challenge Evaluation Plan," https://zenodo.org/record/1199638, 2018.

[19] N. Ryant and et al., "DIHARD Corpus. Linguistic Data Consortium.," 2018.

[20] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B. H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1717–1731, Sept 2010.

[21] Lukas Drude, Jahn Heymann, Christoph Boeddeker, and Reinhold Haeb-Umbach, "NARA-WPE: A Python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing," in *13. ITG Fachtagung Sprachkommunikation (ITG 2018)*, Oct 2018.

[22] Federico Landini, Shuai Wang, Mireia Diez, Lukáš Burget, Pavel Matějka, Kateřina Žmolíková, Ladislav Mošner, Anna Silnova, Oldřich Plchot, Ondřej Novotný, Hossein Zeinali, and Johan Rohdin, "BUT System for DIHARD Speech Diarization Challenge 2019," in *Conference on Acoustics, Speech and Signal Processing (ICASSP) 2020*, 2020.

[23] Mireia Diez, Lukáš Burget, Federico Landini, Shuai Wang, and Honza Černocký, "Optimizing Bayesian HMM based x-vector clustering for the second DIHARD speech diarization challenge," in *Conference on Acoustics, Speech and Signal Processing (ICASSP) 2020*, 2020.

[24] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011, number EPFL-CONF-192584.

[25] Kaldi, "SRE16 v2," https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2, [Downloaded: 2017-12].

[26] Daniel Garcia-Romero and Carol Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proceedings of Interspeech 2011*, 2011.

[27] Jim Pitman, "Exchangeable and partially exchangeable random partitions," *Probability Theory and Related Fields*, vol. 102, pp. 145–158, 1995.

[28] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky, "A sticky hdp-hmm with application to speaker diarization," *The Annals of Applied Statistics*, pp. 1020–1056, 2011.

[29] Aonan Zhang, Quan Wang, Zhenyao Zhu, John Paisley, and Chong Wang, "Fully supervised speaker diarization," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6301–6305.

[30] Themos Stafylakis, Patrick Kenny, Vishwa Gupta, and Pierre Dumouchel, "Compensation for inter-frame correlations in speaker diarization and recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7731–7735.

[31] Patrick Kenny, Douglas Reynolds, and Fabio Castaldo, "Diarization of telephone conversations using factor analysis," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 6, pp. 1059–1070, 2010.

[32] Niko Brümmer, *Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech*, Ph.D. thesis, Stellenbosch University, 2010.

[33] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

A. Discriminative disclaimer

Although our model (1) for embedding extraction is generative, where the embeddings are supposed to have generated the observed speech, we must emphasize that our recipe is discriminative in the sense that the observed data is not fully modelled. Recall (8), reproduced here for convenience:

$$P(s_t \mid \mathbf{x}_t) = h(s_t) \times f(s_t, \mathbf{x}_t) \qquad (20)$$

In the discriminative training of f, h plays no role and it fails to provide a mechanism to ensure that once f has been trained, then (20) will give normalizable distributions for the observed data. Discriminative training does not guarantee that some function $h(s_t) > 0$ exists that would normalize (20) for every $\mathbf{x}_t$:

$$\int h(s_t) f(s_t, \mathbf{x}_t) \, ds_t = 1, \forall \mathbf{x}_t \in \mathbb{R}^D \qquad (21)$$

Arguably the best practical way to (approximately) enforce this constraint, is to do generative rather than discriminative training. This would typically require some *variational Bayes* (VB) recipe to deal with the intractable marginalizations over the hidden variables. In VB, the hidden variable posterior is approximate and in general will also *not* be exactly consistent with a normalizable generative model for the observed data. However, in this case, the training objective (the evidence lower bound, or ELBO) will at least tend to minimize this inconsistency, by effectively minimizing KL divergence from approximate to true posterior [33].

Apart from stating that we *do* plan to try generative VB training in future, we could motivate that the failure of (21) should not trouble us any more than it seems to trouble the designers and users of any other discriminative training recipes—or indeed any recipe that extracts features without a full generative model of the observed data. Discriminative recipes generally do not define generative models for the data and even in cases where they do, discriminatively trained generative models are usually defective as models of the observed data. If we do not need $h(s_t)$ during training or runtime, should it unduly trouble us that it may not exist?