Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

Partial AUC Metric Learning Based Speaker Verification Back-End

Zhongxin Bai, Xiao-Lei Zhang, and Jingdong Chen

Center for Intelligent Acoustics and Immersive Communications and
School of Marine Science and Technology, Northwestern Polytechnical University
zxbai@mail.nwpu.edu.cn, xiaolei.zhang@nwpu.edu.cn, jingdongchen@ieee.org

Abstract

Equal error rate (EER) is a widely used evaluation metric for speaker verification, which reflects the performance of a verification system at a given decision threshold. However, a value of threshold tuned from one application scenario is generally not optimal when the system is used in another scenario. This motivates the need for optimizing the performance at a range of decision thresholds. To fulfill this objective, we propose to optimize the parameters of a squared Mahalanobis distance metric for directly maximizing the partial area under the ROC curve (pAUC) given an interested range of false positive rate. Experimental results on the NIST SRE 2016 and the core tasks of the Speakers in the Wild (SITW) datasets illustrate the effectiveness of the proposed algorithm.

1. Introduction

A speaker verification system usually includes two parts—a front-end [1–6] and a back-end. This paper focuses on the study of back-ends. A back-end is used to examine the similarity of two identity vectors. Common back-ends include cosine similarity scoring [1], probabilistic linear discriminant analysis (PLDA) [7,8], and deep learning based methods [9], most of which minimize surrogate loss functions. However, minimizing surrogate loss functions may not necessarily correspond to the optimal performance under the EER evaluation metric. Therefore, optimizing the EER directly has received much attention. For instance, a cosine metric learning algorithm was developed in [10] to learn a linear transform for minimizing the overlap region of decision scores. Similarly, the work in [11] proposed a triplet loss based cosine similarity metric learning back-end. Besides, some deep learning based algorithms [12, 13] learn an end-to-end mapping function to directly map acoustic features to a decision score.

While it has been widely used to evaluate speaker verification, EER provides only one perspective on performance and is not sufficient to reflect the full picture of a real speaker verification system. For example, the false positive rate (FPR) of a bank security system should be controlled in an extremely low range, e.g., lower than 0.01%; on the contrary, a terrorist detection system for public security is expected to work at a large recall rate range, e.g., higher than 99%, both cases are not characterized by an EER point as illustrated in Fig.1. Motivated by this principle as well as the work in [14, 15], we focus on maximizing the pAUC where the working points locate, as shown in the shadow area of Fig. 1 and propose a Mahalanobis distance metric learning back-end, named *pAUCMetric*, for optimizing the pAUC.

This work contributes the following three aspects to literature as compared to the existing methods [16, 17]. First, we propose to optimize the interested pAUC, while the existing works optimize the full AUC, which can be viewed as a special case of pAUC with $\alpha = 0$ and $\beta = 1$. Second, we propose to optimize pAUC by a Mahalanobis metric learning, which can be combined naturally with widely used front-ends, such as i-vectors and x-vectors. Third, the developed pAUC optimization can be easily extended to an end-to-end optimization.

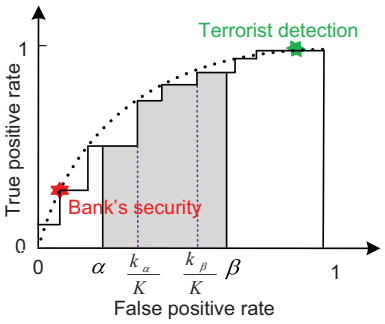

Figure 1: pAUC over a false positive rate range of $[\alpha, \beta]$, where $\alpha \in [0, 1)$ and $\beta \in (0, 1]$.

2. pAUC Metric learning back-end

The proposed pAUCMetric back-end uses the following squared Mahanalobis distance to judge whether two identity vectors $\mathbf{x}_n$ and $\mathbf{y}_n$ belong to the same speaker:

$$S(\mathbf{x}_n, \mathbf{y}_n; \mathbf{M}) = (\mathbf{x}_n - \mathbf{y}_n)^T \mathbf{M} (\mathbf{x}_n - \mathbf{y}_n), \qquad (1)$$

where $\mathbf{M}$ is a symmetric positive semi-definite matrix to be learned by pAUCMetric. For simplicity of exposition, we denote $\mathbf{z}_n = \mathbf{x}_n - \mathbf{y}_n$, and write $S(\mathbf{x}_n, \mathbf{y}_n; \mathbf{M})$ as $S(\mathbf{z}_n; \mathbf{M})$.

2.1. Objective function

Assuming we have a training set $\mathcal{X} = \{\mathbf{x}_{pq} | p = 1, \cdots, P; q = 1, \cdots, P_q\}$, where p and q represent the qth utterance of the pth speaker. The pAUC calculation needs to construct a pairwise set $\mathcal{T} = \{(\mathbf{x}_n, \mathbf{y}_n; l_n) | n = 1, 2, \cdots, N\}$ from $\mathcal{X}$, where l_n is the ground-truth label. If $\mathbf{x}_n$ and $\mathbf{y}_n$ come from the same speaker, then $l_n = 1$; otherwise $l_n = -1$.

According to l_n, we divide $\mathcal{T}$ into two subsets of true and imposter trials respectively, ie. $\mathcal{P} = \{(\mathbf{z}_j^+, l_j = 1) | j = 1, \cdots, J\}$, and $\mathcal{N} = \{(\mathbf{z}_k^-, l_k = -1) | k = 1, \cdots, K\}$. In order to fulfill the FPR constraint of pAUC calculation, we further define a subset of $\mathcal{N}$ as $\mathcal{N}_0 = \{(\mathbf{z}_r^-, l_r = -1) | r = 1, \cdots, R\}$. $\mathcal{N}_0$ is calculated as follows.

- We first replace $[\alpha,\beta]$ by $[k_\alpha/K, k_\beta/K]$ where $k_\alpha = \lceil K\alpha \rceil + 1$ and $k_\beta = \lfloor K\beta \rfloor$ are two integers.

10.21437/Odyssey.2020-53

- Then, $\{S(\mathbf{z}_k^-;\mathbf{M})\}_{\mathbf{z}_k^- \in \mathcal{N}}$ are sorted in ascending order.
- Finally, we pick the samples ranked from the top k_αth to k_βth positions to form $\mathcal{N}_0$.

Then, pAUC can be computed as follows,

$$\begin{aligned}\text{pAUC} =& 1 - \frac{1}{JR}\sum_{j=1}^{J}\sum_{r=1}^{R}\big[\mathbb{I}(S(\mathbf{z}_j^+;\mathbf{M}) > S(\mathbf{z}_r^-;\mathbf{M})) \\ &+ \frac{1}{2}\mathbb{I}(S(\mathbf{z}_j^+;\mathbf{M}) = S(\mathbf{z}_r^-;\mathbf{M}))\big] \end{aligned} \quad (2)$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the statement is true, and 0 otherwise. However, directly optimizing (2) is an NP-hard problem. To circumvent this, we relax (2) by replacing the indicator function by a hinge loss function:

$$\begin{aligned}&\ell_{\text{hinge}}(S(\mathbf{z}_j^+;\mathbf{M}) > S(\mathbf{z}_r^-;\mathbf{M})) = \\ &\max\left[0, \delta - \big(S(\mathbf{z}_r^-;\mathbf{M}) - S(\mathbf{z}_j^+;\mathbf{M})\big)\right]\end{aligned} \quad (3)$$

where $\delta > 0$ is a tunable hyper-parameter. Substituting (3) into (2) and further changing the maximization problem (2) into an equivalent minimization one gives (4).

$$\ell = \frac{1}{JR}\sum_{j=1}^{J}\sum_{r=1}^{R}\max\Big(0, \delta - S(\mathbf{z}_r^-;\mathbf{M}) + S(\mathbf{z}_j^+;\mathbf{M})\Big) \quad (4)$$

To prevent from overfitting to the training data, we add a regularization term $\lambda\Omega(\cdot)$ to the minimization problem. So, our optimization problem becomes

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \ell(\mathcal{P}, \mathcal{N}; \mathbf{M}) + \lambda\Omega(\mathbf{M}), \quad (5)$$

where $\lambda\Omega(\cdot)$ is defined as:

$$\lambda\Omega(\mathbf{M}) = \frac{\gamma}{J}\sum_{j=1}^{J} S(\mathbf{z}_j^+;\mathbf{M}) + \mu[\text{tr}(\mathbf{I}_0^{-1}\mathbf{M}) - \text{logdet}(\mathbf{M})] \quad (6)$$

with γ and μ being two tunable hyper-parameters and $\mathbf{I}_0$ being an identity matrix. The first one $\frac{1}{J}\sum_{j=1}^{J} S(\mathbf{z}_j^+;\mathbf{M})$ [18] aims to bound $S(\mathbf{z}_j^+;\mathbf{M})$ in (4). The second term $\text{tr}(\mathbf{I}_0^{-1}\mathbf{M}) - \text{logdet}(\mathbf{M})$ [19] is used to improve the generalization ability and further constrain $\mathbf{M}$ to be positive semi-definite.

2.2. Optimization algorithm

In order to solve the optimization problem in (5), we first define an index matrix $\mathbf{\Pi} \in \{0,1\}^{J\times R}$:

$$\mathbf{\Pi}(j,r) = \begin{cases} 1, & \text{if } \delta + S(\mathbf{z}_j^+;\mathbf{M}) > S(\mathbf{z}_r^-;\mathbf{M}) \\ 0, & \text{otherwise} \end{cases}. \quad (7)$$

Substituting (7) into (5) gives

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \langle \mathbf{P} + \gamma\mathbf{P}_{\mathcal{P}}, \mathbf{M}\rangle_F + \mu\left[\text{tr}(\mathbf{I}_0^{-1}\mathbf{M}) - \text{logdet}(\mathbf{M})\right] \quad (8)$$

where $\langle\cdot\rangle_F$ denotes the Frobenius norm operator, and

$$\mathbf{P}_{\mathcal{P}} = \frac{1}{J}\sum_{j=1}^{J}\mathbf{z}_j^+\mathbf{z}_j^{+T}, \quad (9)$$

$$\mathbf{P} = \frac{1}{JR}\sum_{j=1}^{J}\sum_{r=1}^{R}\mathbf{\Pi}(j,r)(\mathbf{z}_j^+\mathbf{z}_j^{+T} - \mathbf{z}_r^-\mathbf{z}_r^{-T}). \quad (10)$$

We employ the proximal point algorithm (PPA) [14] to optimize (8). The algorithm, which is summarized in Algorithm 1, consists of the following three steps at each iteration:

Algorithm 1 Proximal point algorithm.

Input: Development sets $\mathcal{X}$, and batch size s; Hyper-parameter δ; Regularization parameters $\gamma > 0, \mu > 0$; Step size $\eta > 0$, and $\lambda = \eta\mu$; False positive rate: $\alpha \geq 0, \beta > 0$;
Output: The optimal matrix $\mathbf{M}^*$

```
t ← 0
repeat
    Construct T^t
    Calculate P^t, N^t and N_0^t
    Calculate P^t and P_P^t on P^t and N_0^t by (9) and (10)
    Update M_{t+1} ← φ_λ^+(M_t − η(P^t + γP_P^t + μI_0))
    t ← t + 1
until convergence
```

- We construct a pairwise set $\mathcal{T}^t$ at each iteration by a random sampling strategy as follows. We first randomly select s speakers from $\mathcal{X}$, then randomly select two identity vectors from each of the selected speakers, and finally construct $\mathcal{T}^t$ by a full permutation of the $2s$ identity vectors. $\mathcal{T}^t$ contains s true training trials and $s(2s-1)) - s$ imposter training trials.
- The second step firstly calculates $\mathcal{P}^t$, $\mathcal{N}^t$ and $\mathcal{N}_0^t$, then computes $\mathbf{P}^t$ and $\mathbf{P}_{\mathcal{P}}^t$ according to (9) and (10).
- The third step updates $\mathbf{M}$ by PPA [14], which first applies eigenvalue decomposition to $\mathbf{X} = \mathbf{M}_t - \eta(\mathbf{P}^t + \gamma\mathbf{P}_{\mathcal{P}}^t + \mu\mathbf{I}_0)$, i.e. $\mathbf{X} = \mathbf{U}\mathbf{V}\mathbf{U}^T$ where $\mathbf{V} = \text{diag}([v_1, v_2, \cdots, v_d])$ with $v_1 \geq v_2 \geq \cdots \geq v_d$, and then adopts the following updating equation:

$$\phi_\lambda^+(\mathbf{x}) = \mathbf{U}\text{diag}([\phi_\lambda^+(v_1), \cdots, \phi_\lambda^+(v_d)])\mathbf{U}^T, \quad (11)$$

where $\phi_\lambda^+(v) = \left[(v^2 + 4\lambda)^{1/2} + v\right]/2$, and d is the dimension of the input feature.

The pAUCMetric can take any speaker features as its input, such as i-vectors or x-vectors. Here we use the latent variable of the two-covariance-based PLDA [20] as its input.

3. Experiments

We followed the kaldi recipes [21] of *"kaldi-master/egs/sre16/v1"* and *"kaldi-master/egs/sre16/v2"* to evaluate our proposed algorithm. The training data consists of Switchboard (SWBD) and NIST speaker recognition evaluation (SRE) database. SWBD consists of Switchboard Cellular 1 and 2 as well as Switchboard 2 Phase 1, 2, and 3. It contains 28,181 English utterances from 2,594 speakers. The SRE database consists of NIST SREs from 2004 to 2010 along with Mixer 6. It contains 64,388 telephone and microphone recordings from 4,392 speakers. In this paper, we adopted the same data augmentation scheme as in [5] to further increase the amount and diversity of the training data. We only used the augmented SRE data to train the back-ends.

The evaluation data include the Cantonese portion of the NIST SRE 2016 [22] and the core tasks of the Speakers in the Wild (SITW) dataset [23]. The Cantonese contains 965,393 trials. The SITW dataset has two evaluation tasks—Dev.Core and Eval.Core, which consist of 338,226 and 721,788 trails respectively.

To investigate the performance of the proposed back-end, we combine it with the GMM-UBM/i-vector [1] and x-vector [5] front-ends respectively. The front-ends are implemented by Kaldi [21]. We compare pAUCMetric with the PLDA:

Table 1: Performance comparison of PLDA and pAUCMetric on the Cantonese dataset of the NIST SRE 2016 corpus. The terms "adaptation"/"no-adaptation" denote the training situations with/without the domain adaptation technique.

		i-vector				x-vector			
	Back-ends	EER (%)	DCF10^{-2}	$\text{pAUC}_{[0,0.01]}$	AUC	EER (%)	DCF10^{-2}	$\text{pAUC}_{[0,0.01]}$	AUC
No-adaptation	PLDA	10.29	0.6549	0.5703	0.9638	6.78	0.5311	0.6892	0.9821
	pAUCMetric	**9.53**	**0.6493**	**0.5783**	**0.9685**	**6.00**	**0.5033**	**0.7173**	**0.9855**
	Relative improvement	*7.39%*	*0.86 %*	*1.86%*	*12.98%*	*11.50%*	*5.23%*	*9.04%*	*18.99%*
Adaptation	PLDA	8.89	0.5976	0.6242	0.9701	4.82	0.3994	0.8008	0.9904
	pAUCMetric	**7.93**	**0.5772**	**0.6465**	**0.9766**	**4.19**	**0.3788**	**0.8182**	**0.9925**
	Relative improvement	*10.80%*	*3.41 %*	*5.93%*	*21.74%*	*13.07%*	*5.16%*	*8.73%*	*21.88%*

Table 2: Performance comparison of PLDA and pAUCMetric on the core tasks of SITW corpus.

		i-vector				x-vector			
	Back-ends	EER (%)	DCF10^{-2}	$\text{pAUC}_{[0,0.01]}$	AUC	EER (%)	DCF10^{-2}	$\text{pAUC}_{[0,0.01]}$	AUC
Dev.Core.	PLDA	9.20	0.6189	0.5903	0.9719	6.85	0.5135	0.6972	0.9848
	pAUCMetric	**8.73**	**0.6056**	**0.5989**	**0.9744**	**5.91**	**0.4999**	**0.7240**	**0.9880**
	Relative improvement	*5.11%*	*2.15 %*	*2.10%*	*8.90%*	*13.72%*	*2.65%*	*8.85%*	*21.05%*
Eval.Core.	PLDA	10.03	0.6460	0.5630	0.9685	6.75	0.5458	0.6746	0.9841
	pAUCMetric	**9.65**	**0.6392**	**0.5706**	**0.9696**	**6.10**	**0.5276**	**0.7022**	**0.9861**
	Relative improvement	*3.79%*	*1.05 %*	*1.74%*	*3.49%*	*9.63%*	*3.33%*	*8.48%*	*12.58%*

- **PLDA:** When the i-vector front-end is used, we apply LDA to reduce the dimension of the i-vectors from 600 to 200, which is a typical parameter setting. When the x-vector front-end is used, we follow the parameter setting in [5], which reduces the dimension of the x-vectors from 512 to 150. Finally, the output of the LDA is taken as the input of PLDA for verification.
- **pAUCMetric:** The same LDA as the above baseline is used, and then we extract the latent variable of the above PLDA as the input of pAUCMetric. The default hyperparameters of pAUCMetric are as follows: $\alpha = 0$, $\beta = 0.01$, $\gamma = 0.5$, $\mu = 10^{-3}$, $\eta = 10$, and $s = 500$. We will show that pAUCMetric is insensitive to the hyperparameter selection in Section 3.2.

Because most of the training data are in English while the evaluation data of NIST SRE 2016 are in Cantonese, we conduct domain adaptation by using an unlabeled major dataset in NIST SRE 2016, which consists of 2,272 utterances. The adaptation technique is implemented in "*kaldi-master/egs/sre16/v2*" of Kaldi.

The evaluation metrics used include EER, the minimum detection cost function with $P_{\text{target}} = 10^{-2}$ (DCF10^{-2}), the pAUC with $\alpha = 0$ and $\beta = 0.01$ ($\text{pAUC}_{[0,0.01]}$), and AUC.

3.1. Main results:

Table 1 lists the comparison results on the Cantonese test data. It is seen that pAUCMetric yields better performance than PLDA, given both the i-vector and x-vector front-ends. Specifically, pAUCMetric obtains approximately 9% and 20% relative improvement over PLDA in terms of $\text{pAUC}_{[0,0.01]}$ and AUC, respectively. Moreover, it achieves more than 11% relative EER reduction and 5% relative DCF10^{-2} reduction over PLDA.

Table 2 lists the results on the core tasks of the SITW corpus. It is seen from this table that pAUCMetric yields better performance than PLDA. Specifically, when the x-vector front-end is used, pAUCMetric achieves more than 8% relative $\text{pAUC}_{[0,0.01]}$ improvement over PLDA; it also obtains more than 20% and 12% relative AUC improvement on the Dev.Core and Eval.Core tasks respectively; Moreover, it achieves 10% relative EER reduction and 3% relative DCF10^{-2} reduction. Although the performance improvement with the i-vector front-end is not so significant as that with the x-vector front-end, the trends with different front-ends are consistent.

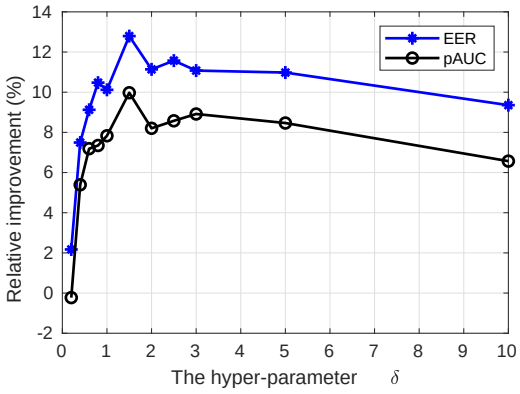

Figure 2: Relative performance improvement of pAUCMetric over PLDA with respect to δ in the x-vector space on the Cantonese dataset without domain adaptation.

3.2. Effects of the hyperparameters of pAUCMetric:

The reason why we set $\alpha = 0$ and $\beta = 0.01$ is that the number of the imposter trials is much larger than that of the true trials, hence restricting the working area to a small FPR region makes the algorithm focus on discriminating the difficult trials.

We adopted grid search to study the impact of the values of δ, γ, and μ on performance in the x-vector space where no adaptation was adopted. Specifically, we searched δ in $[0, 10]$ and set the other hyperparameters to their default values. Figure 2 shows the relative performance improvement over PLDA on the Cantonese dataset. From the figure, we find that δ is robust in a wide range with the best δ around 1.5. We searched γ and μ in grid jointly as listed in Tables 3 and 4 with the other hyperparameters set to their default values. It is observed that the stable working region is $\mu \in [0, 10^{-3}] \cap \gamma \in [0, 1.5]$. Inter-

Table 3: Relative EER reduction of pAUCMetric over PLDA with respect to γ and μ in the x-vector space on the Cantonese dataset without domain adaptation.

γ \ μ	0	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0	7.5	7.4	7.8	6.8	6.5	7.6	1.2	-5.9
0.01	7.8	8.8	7.8	7.5	7.6	7.6	1.7	-6.2
0.10	9.4	7.4	9.4	9.1	9.3	10.2	1.7	-6.0
0.50	9.6	9.5	9.6	10.8	10.5	11.0	1.9	-6.2
1.00	8.0	10.6	8.0	9.5	8.5	10.9	2.1	-6.0
1.50	6.9	5.5	6.8	8.7	8.4	8.6	2.7	-5.8
3.00	-1.4	-3.3	-1.3	-3.6	-0.2	1.0	1.1	-5.6

Table 4: Relative $\mathrm{pAUC}_{[0,0.01]}$ reduction of pAUCMetric over PLDA with respect to γ and μ in the x-vector space on the Cantonese dataset without domain adaptation.

γ \ μ	0	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0	5.0	4.6	4.9	4.2	4.1	4.3	-0.6	-6.2
0.01	5.2	5.4	5.2	5.4	4.9	4.8	-0.6	-6.3
0.10	6.9	5.4	6.9	6.6	6.6	7.1	-0.1	-6.1
0.50	7.3	7.4	7.3	7.3	7.4	8.0	-0.2	-6.1
1.00	6.1	7.3	6.1	6.6	4.9	7.5	0.4	-6.1
1.50	4.6	3.8	4.6	2.9	6.0	5.6	1.5	-5.9
3.00	-3.5	-4.4	-1.8	-2.6	-3.2	-0.3	-0.7	-6.1

estingly, pAUCMetric still works well even without regularization, i.e. $\mu = 0$ and $\gamma = 0$. The above observation is consistent across all training scenarios of this paper.

Finally, the performance of pAUCMetric is also insensitive to the batch size s. Due to the length limitation of this paper, we will not report the result here. In conclusion, pAUCMetric is found insensitive to the values of those 6 hyperparameters .

4. Conclusion

In this paper, a Mahalanobis metric learning back-end is proposed to maximize pAUC for speaker verification. Because directly optimizing pAUC is an NP-hard problem, we first relaxed the optimization problem to a polynomial-time solvable one, and then adopted a random sampling strategy to reduce the computational complexity. Experiments were carried out and the results on the Cantonese dataset of NIST 2016 SRE and the core tasks of SITW demonstrated the effectiveness of the pAUCMetric back-end. Experiments also showed that pAUCMetric is insensitive to the values of the hyperparameters for the studied input features and training scenarios.

5. Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2018AAA0102200 and in part by the Key Program of National Science Foundation of China (NSFC) Grant No. 61831019 and the NSFC and Israel Science Foundation (ISF) joint research program under Grant No. 61761146001, in part by the NSFC funding scheme under grant No. 61671381, in part by the Project of the Science, Technology, and Innovation Commission of Shenzhen Municipality under grant No. JCYJ20170815161820095, and in part by the Shaanxi Natural Science Basic Research Program under grant No. 2018JM6035.

6. References

[1] N. Dehak, P. J. 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, 2011.

[2] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1695–1699.

[3] P. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proc. Odyssey*, 2014, pp. 293–298.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[6] N. Li, D. Tuo, D. Su, Z. Li, and D. Yu, "Deep discriminative embeddings for duration robust speaker verification," in *Proc. Interspeech 2018*, 2018, pp. 2262–2266.

[7] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, 2010, p. 14.

[8] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[9] O. Ghahabi and J. Hernando, "Deep belief networks for i-vector based speaker recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1700–1704.

[10] Z. Bai, X.-L. Zhang, and J. Chen, "Cosine metric learning for speaker verification in the i-vector space," in *Proc. Interspeech 2018*, 2018, pp. 1126–1130.

[11] S. Novoselov, V. Shchemelinin, A. Shulipa, A. Kozlov, and I. Kremnev, "Triplet loss based cosine similarity metric learning for text-independent speaker recognition," in *Proc. Interspeech 2018*, 2018, pp. 2242–2246.

[12] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[13] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5115–5119.

[14] J. Huo, Y. Gao, Y. Shi, and H. Yin, "Cross-modal metric learning for auc optimization," *IEEE Transactions on Neural Networks and Learning Systems, PP (99)*, pp. 1–13, 2018.

[15] T. Takenouchi, O. Komori, and S. Eguchi, "An extension of the receiver operating characteristic curve and auc-optimal classification," *Neural Computation*, vol. 24, no. 10, pp. 2789–2824, 2012.

[16] L. P. Garcia-Perera, J. A. Nolazco-Flores, B. Raj, and R. Stern, "Optimization of the det curve in speaker verification," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 318–323.

[17] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Optimization of the area under the roc curve using neural network supervectors for text-dependent speaker verification," *arXiv preprint arXiv:1901.11332*, 2019.

[18] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[19] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 209–216.

[20] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision*. Springer, 2006, pp. 531–542.

[21] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[22] "Nist 2016 speaker recognition evaluation plan," https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016, 2016.

[23] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (sitw) speaker recognition database." in *Interspeech*, 2016, pp. 818–822.