REAL ADDITIVE MARGIN SOFTMAX FOR SPEAKER VERIFICATION

Lantian Li, Ruiqian Nai, Dong Wang

Center for Speech and Language Technologies, BNRist, Tsinghua University, China

ABSTRACT

The additive margin softmax (AM-Softmax) loss has delivered remarkable performance in speaker verification. A supposed behavior of AM-Softmax is that it can shrink within-class variation by putting emphasis on target logits, which in turn improves margin between target and non-target classes. In this paper, we conduct a careful analysis on the behavior of AM-Softmax loss, and show that this loss does not implement real max-margin training. Based on this observation, we present a *Real* AM-Softmax loss which involves a true margin function in the softmax training. Experiments conducted on VoxCeleb1, SITW and CNCeleb demonstrated that the corrected AM-Softmax loss consistently outperforms the original one. The code has been released at *https://gitlab.com/csltstu/sunine*.

Index Terms— speaker verification, additive margin softmax, max-margin metric learning

1. INTRODUCTION

A key concept of modern speaker verification techniques is to *embed* a variable-length speech utterance into a fixed-length dense vector, usually called *speaker embedding*. By utilizing the strength of deep neural nets (DNNs) in learning task-oriented features, this embedding process collects speaker-related information and eliminates other nuances, leading to highly discriminative representations for speaker traits.

Early research borrows the experience from speech recognition and employs the classification framework that employs softmax as the activation and cross-entropy as the training objective [1]. By this so-called *softmax training*, vectors of utterances from the same speaker are put together and those from different speakers are repulsed from each other. Theoretically, this objective is optimal for discriminating the speakers in the *training* set, but the performance on unseen speakers is not guaranteed. Unfortunately, speaker verification is an open-set problem in nature, and handling novel speakers is a primary request.

This work was supported by the National Natural Science Foundation of China under Grants No.61633013 and No.62171250, and also the Tsinghua-SPD Bank Joint Research Center for Digital Finance Technologies. Dong Wang is the corresponding author (wangdong99@mails.tsinghua.edu.cn).

A wealth of research has been conducted to tackle this problem. One direction is to learn a metric space for speakers [2], by either pair-based distance [3], triplet-based distance [4, 5], or group-based distance [6, 7, 8]. This *metric learning* offers an elegant solution for the open-set problem. However, the training requires large amounts of pairs or triples, which are not easy to design [9]. Moreover, the training signal produced by pairs or triplets is local and weak, leading to ineffective training [10, 11].

Due to the problems associated with metric learning, many studies stick on softmax training, but try to remedy its potential weakness on unknown speakers. The first approach is to involve additional regularization to reduce the within-speaker variation, for example via the central loss [12, 13], ring loss [14] or Gaussian constraint [15]. Another approach is to employ various normalization methods to enforce the speaker vectors following some desired distributions, in particular Gaussian. This can be conducted either globally or class-dependently. The former is represented by the famous length normalization [16] and the latter is represented by the deep normalization model based on normalization flows [17].

Perhaps the most prominent approach is the various max-margin softmax training methods. Max-margin training aims to maximize the discrepancy between target pair distance and non-target pair distance, and has been extensively used in metric learning, e.g., [4, 10]. This idea was adopted to improve softmax training, leading to various max-margin softmax loss, including Additive Margin Softmax (AM-Softmax) [18, 19] and Additive Angular Margin Softmax (AAM-Softmax) [20]. The intuition of these max-margin losses is to make intra-class attraction a more tough task than inter-class repulsion, so that the target class and non-target class are better separated. In the setting of softmax training, this means that the difference between target logits and non-target logits should be maximized.

Although the effectiveness has been widely demonstrated, we will show in this paper that these max-margin softmax losses do not implement a true margin on target and non-target logits. Specifically, margin, according to the canonical definition, is not only the distance of two classes, but the region where the loss is incurred [21]. In metric learning based on triplets, this is often formulated by the following form [4, 5]:

$$\mathcal{L}_{\text{margin}} = \max(0, d_p - d_n + m) \tag{1}$$

where d_p and d_n represent the distance of positive and negative pairs, respective, and m is the margin. According to this definition, the loss is incurred only by hard negative pairs, i.e., negative pairs that are similar to positive pairs. The margin defined in the max-margin losses mentioned above ignores the max operation, so does not hold such property.

In this study, we propose a simple modification to AM-Softmax to enforce the true max-margin training. With this change, the non-target logits will not contribute loss if it has been much less than the target logit, so that the model will save the parameters to handle other harder non-target logits. We choose AM-Softmax to conduct the analysis because its form is simple and has obtained good performance in literature and our own experiments, though the analysis can be similarly conducted on AAM-Softmax and other margin-based softmax losses.

2. METHODS

In this section, we firstly revisit the AM-Softmax loss and discuss its properties and potential limitation, which is followed by the presentation of our Real AM-Softmax.

2.1. AM-Softmax

As a classification loss for training the speaker discriminative DNNs, the additive margin Softmax (AM-Softmax) loss [19] can be formulated as:

$$\mathcal{L}_{\text{AM-Softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{s(\cos(\theta_{j,i}))}} \tag{2}$$

where N is the batch size, m denotes the additive margin, s is the scaling factor for training stability. i indexes the training samples, and y_i is the label of the i-th sample. $\theta_{j,i}$ is the angle between the vector of the i-th sample x_i and the representative vector of class j denoted by w_j, therefore:

$$\cos(\theta_{j,i}) = \frac{x_i^T w_j}{||x_i||\,||w_j||} \tag{3}$$

When implemented by DNN, x_i can be read from the activation of the penultimate hidden layer (before the last affine transform) and the weights linked to the j-th output unit can be regarded as w_j. Usually both x_i and w_j are constrained to be unit-length vectors, so that the cosine distance is computed by simply dot product and implemented as matrix multiplication. In this case, $\cos(\theta_{j,i})$ is just the logit (i.e., activation before softmax) on the j-th output unit aroused by the i-th sample.

2.2. Margin factor does not boost margin

Intuitively, due to the margin factor m, the AM-softmax enlarges the margin between target logits and non-target logits, hence leading to a more discriminative model. The reasoning is as follows: in order to get the same loss as the standard softmax, the target logit in AM-Softmax must get higher superiority compared to non-target logits. However, a careful analysis shows this is not exactly the case.

We first make a simple reformulation to the AM-softmax loss as follows:

$$\begin{aligned}\mathcal{L}_{\text{AM-Softmax}} &= \frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{s(\cos(\theta_{j,i}))}}{e^{s(\cos(\theta_{y_i,i})-m)}} \\ &= \frac{1}{N}\sum_{i=1}^{N}\log\Big\{1+\sum_{j\neq y_i}e^{-s(\cos(\theta_{y_i,i})-\cos(\theta_{j,i})-m)}\Big\}\end{aligned} \tag{4}$$

The exponential part of the second term is the most interesting, and it shows that to reduce the loss, the model should enlarge the difference between the target logits and the non-target logits. This will indeed improve the margin; however, this is not the unique property of AM-Softmax, as the standard softmax can be reformulated in the same way and the margin is maximized without difference.[1]

Particularly, it seems that the margin factor m is *not* really related to the margin between the target logits and non-target logits, and setting a larger m will not enforce a larger margin, as it is supposed to do. This can be seen clearly by writing the loss in the following form:

$$\mathcal{L}_{\text{AM-Softmax}} = \frac{1}{N}\sum_{i=1}^{N}\log\Big\{1+e^{sm}\sum_{j\neq y_i}e^{-s(\cos(\theta_{y_i,i})-\cos(\theta_{j,i}))}\Big\} \tag{5}$$

It shows that the contribution of m may change the shape of the loss landscape, but not its form. In particular, it is not directly related to the margin between the target and non-target logits, as defined by $\cos(\theta_{y_i,i}) - \cos(\theta_{j,i})$.

To make it more clear, let us divide the training data into an easy set and a hard set. We shall also set $s = 1$ to expose the contribution of the margin factor m.[2] Our analysis starts from $m = 0$, the special case of the standard softmax. For the samples in the easy set, the target logit dominates the denominator in Eq.(2), and we have:

$$\frac{e^{(\cos(\theta_{y_i,i})-m)}}{e^{(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i}e^{(\cos(\theta_{j,i}))}} \approx 1 \tag{6}$$

[1]Note that setting $s = 1$ and $m = 0$ in Eq.(4), we can recover the standard softmax.

[2]Note this will not limit our discussion as m only appears in the factor e^{sm}. Since s and m are two free parameters, we can always define a new m to absorb s.

This leads to:

$$1 + e^{m} \sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))} \approx 1 \tag{7}$$

The loss associated with this sample can be approximated by:

$$\begin{aligned} & \log\{1 + e^{m} \textstyle\sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))}\} \\ \approx \quad & e^{m} \textstyle\sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))} \end{aligned} \tag{8}$$

This means when m is increased from 0, the contribution of easy samples will be emphasized.

For samples in the hard set, the target logit is weak, so we have:

$$\frac{e^{(\cos(\theta_{y_i,i}) - m)}}{e^{(\cos(\theta_{y_i,i}) - m)} + \sum_{j \neq y_i} e^{s(\cos(\theta_{j,i}))}} \ll 1 \tag{9}$$

This means:

$$1 + e^{m} \sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))} \gg 1 \tag{10}$$

Then the loss associated with this sample is approximated as follows:

$$\begin{aligned} & \log\{1 + e^{m} \textstyle\sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))}\} \\ \approx \quad & m + \log \textstyle\sum_{j \neq y_i} e^{-(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}))} \end{aligned} \tag{11}$$

This form means that setting any m will not change the optimum.

Overall, setting a large m has the effect of boosting the contribution of easy samples, while keeping hard samples unweighted. Certainly, this is not a good property as hard samples are always more concerning. Fortunately, the negative effect cannot be very disastrous. This is because if m is set overlarge, the approximation Eq.(8) for easy samples will be invalid and turns to the form Eq.(11) of hard samples, hence losing its impact.

Interestingly, the scale factor s seems more effective in boosting the margin. If we set $s > 1$, then the contribution of the non-targets j will be boosted if $\cos(\theta_{y_i,i}) - \cos(\theta_{j,i})$ is small, i.e., its contribution will be amplified. This coincides the idea of hard negative pair mining, widely adopted in metric learning [10].

2.3. Real AM-Softmax

We analyzed the property of AM-Softmax, and found that the margin factor m does not play the role of maximizing the margin between target and non-target logits. Although the scale factor s takes the role partly, the *margin* here is not precise. It refers to *discrimination*, rather than the canonical definition by Vapnik et al. [21], as shown in Eq.(1). In this paper, we call this 'intuitive' definition of margin in AM-Softmax as *trivial margin*, and the canonical definition of Eq.(1) as *real margin*. The major difference between these two definitions is that the trivial margin omits the max operation. According to the analysis in the previous section, this omission leads to serious consequence: in fact it shuns true max-margin training.

We will follow the definition of real margin to modify the AM-Softmax loss, and call it *Real* AM-Softmax (RAM-Softmax). This is achieved by slightly modifying the loss function of Eq.(4) as follows:

$$\mathcal{L}_{\text{RAM-Softmax}} = \frac{1}{N} \sum_{i=1}^{N} \log\{1 + \sum_{j \neq y_i} e^{\max\{0, -s(\cos(\theta_{y_i,i}) - \cos(\theta_{j,i}) - m)\}}\} \tag{12}$$

According to this modified formulation, if the target logit is larger than non-target logits by more than m, the loss will be zero, otherwise a positive loss will be incurred. This will encourage the model to focus on hard non-target logits (negative speaker class), and forget easy non-targets that have been well separated. Therefore, the new loss can cannot be reduced by over-training on easy non-target logits; attention must be paid on hard non-target logits. This will balance the contribution of all classes, which arguably alleviates the discrepancy between softmax training and the open-set nature in speaker verification. Moreover, (real) max-margin training enjoys a solid theoretical advantage in model generalization, as shown by Vapnik [21].

Interestingly, if we focus on the exponential part of the RAM-Softmax, the loss is quite similar to the triple loss in metric learning, except that all the non-target logits are treated as negative pairs. RAM-Softmax therefore can be regarded as a graft of softmax training and metric learning.

3. EXPERIMENTS

We will compare the AM-Softmax and Real AM-Softmax on several speaker verification tasks. Note that our purpose is not a SOTA performance; instead, we focus on testing if the 'correct' margin works and how it behaves.

3.1. Data

Three datasets were used in our experiments: VoxCeleb [22], SITW [23] and CNCeleb [24]. More information about these three datasets is presented below.

VoxCeleb: A large-scale audiovisual speaker dataset collected by the University of Oxford, UK. In our experiments, the development set of VoxCeleb2 was used to train the x-vector models, which contains 5,994 speakers in total and entirely disjoints from the VoxCeleb1 and SITW datasets. No data augmentation was used. *VoxCeleb1* was used as the development set to tune the hyperparameters, and *VoxCeleb1-E* and *VoxCeleb1-H* were used to test the performance. Note

Table 1. EER(%) results on VoxCeleb1 and SITW.

Objective	Hyperparameters	VoxCeleb1	VoxCeleb1-H	VoxCeleb1-E	SITW.Dev.Core	SITW.Eval.Core
AM-Softmax	m = 0.20, s = 30	**1.739**	2.895	1.724	**2.811**	3.362
Real AM-Softmax	m = 0.20, s = 30	1.872	3.068	1.883	3.466	3.718
	m = 0.25, s = 30	1.819	2.914	1.781	3.350	3.554
	m = 0.30, s = 30	1.755	**2.812**	**1.696**	3.003	3.417
	m = 0.35, s = 30	1.808	2.888	1.747	2.849	**3.335**

that the pairs of *VoxCeleb1-H* are drawn from identities with the same gender and nationality, hence harder to verify than those in *VoxCeleb1-E*.

SITW: A widely-used evaluation dataset, consisting of 299 speakers. In our experiments, *Dev.Core* was used for parameter tuning and *Eval.Core* was used for testing.

CNCeleb: A large-scale speaker dataset collected by Tsinghua University. It contains more than 600k utterances from 3,000 Chinese celebrities, and the utterances cover 11 different genres, therefore much more challenging than VoxCeleb1 and SITW. The entire dataset was split into two parts: *CNCeleb.Train*, which involves 2,800 speakers and was used to train the x-vector models; *CNCeleb.Eval*, which involves 200 speakers, was used for testing.

3.2. Settings

We follow the x-vector architecture [25] to construct the speaker embedding model, which accepts 80-dimensional Fbanks as input features and adopts the ResNet34 topology for frame-level feature extraction. Statistical pooling strategy is employed to construct utterance-level representations. These representations are then transformed by a fully-connected layer to generate logits and are fed to a softmax layer to generate posterior probabilities over speakers. Once trained, the 512-dimensional activations of the last fully-connected layer are read out as an x-vector. The simple cosine distance is used to score the trials in our experiments.

3.3. Results on VoxCeleb1 and SITW

In the first experiment, we trained the x-vector models using VoxCeleb2, and then test the performance on VoxCeleb1 and SITW. The results in terms of equal error rate (EER) are shown in Table 1. Note that the hyperparameters m and s of AM-Softmax have been carefully tuned, so the performance can be regarded as optimized. To put focus on margin m, we chose the same s in Real AM-Softmax.

Firstly, it can be observed that Real AM-Softmax offers better performance than AM-Softmax in VoxCeleb1-H, VoxCeleb1-E and SITW.Eval.Core trials, when m was chosen according to the performance on their own development set. This improvement is not very remarkable but consistent, demonstrating that the real margin is a correct modification. In particular, since s is optimized for AM-Softmax, the improvement obtained with real margin can be regarded believable. Note that the training process with Real AM-Softmax is more stable. For example, Real AM-Softmax can use a larger margin m and converges much faster (15-20 epoches) than AM-Softmax (40-50 epoches) in the VoxCeleb dataset.

3.4. Results on CNCeleb

The results in Table 1 indicate that Real AM-Softmax is more superior under challenging test conditions. For instance, the performance improvement with Real AM-Softmax is more significant on VoxCeleb1-H than on VoxCeleb1-E.

To verify this conjecture, we designed an experiment on CNCeleb, the more challenging dataset. The x-vector models were trained with CNCeleb.Train, and were tested on CNCeleb.Eval. Again, the parameters of AM-Softmax were optimized, and we chose the same s in Real AM-Softmax.

Table 2 shows the EER results with various settings of m for Real AM-Softmax. It can be observed that Real AM-Softmax outperforms AM-Softmax on this more challenging dataset, and the relative improvement is slightly more significant than on VoxCeleb1 and SITW.

Table 2. EER(%) results on CNCeleb.

Objective	Hyperparameters	CNCeleb.Eval
AM-Softmax	m = 0.10, s = 30	11.450
Real AM-Softmax	m = 0.10, s = 30	11.618
	m = 0.15, s = 30	11.323
	m = 0.20, s = 30	**11.049**
	m = 0.25, s = 30	11.422

4. CONCLUSIONS

In this paper we analyze the popular AM-Softmax loss and identify its inability to conduct real max-margin training. Based on this analysis, we present a real AM-Softmax loss following the canonical max-margin definition. With this new loss, hard negative samples are supposed to be more attended. We tested it on VoxCeleb1, SITW and CNCeleb, and obtained marginal but consistent performance improvement. These results demonstrate that the modified real margin function is valid. Further gains are expected by tuning the scale and margin factors freely. Moreover, we conjecture AAM-Softmax and other margin-based softmax losses can be improved in the same way as they both define trivial margin as AM-Softmax.

5. REFERENCES

[1] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*, 2018, pp. 5329–5333.

[2] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee-Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han, "In defence of metric learning for speaker recognition," in *INTERSPEECH*, 2020, pp. 2977–2981.

[3] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*. IEEE, 2016, pp. 5115–5119.

[4] Lantian Li, Dong Wang, Chao Xing, and Thomas Fang Zheng, "Max-margin metric learning for speaker recognition," in *ISCSLP*, 2016, pp. 1–4.

[5] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[6] Jixuan Wang, Kuan-Chieh Wang, Marc T Law, Frank Rudzicz, and Michael Brudno, "Centroid-based deep metric learning for speaker recognition," in *ICASSP*. IEEE, 2019, pp. 3652–3656.

[7] Yuheng Wei, Junzhao Du, and Hui Liu, "Angular margin centroid loss for text-independent speaker recognition.," in *INTERSPEECH*, 2020, pp. 3820–3824.

[8] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *ICASSP*. IEEE, 2018, pp. 4879–4883.

[9] Anurag Chowdhury and Arun Ross, "Deepvox: Discovering features from raw audio for speaker recognition in degraded audio signals," *arXiv preprint arXiv:2008.11668*, 2020.

[10] Hervé Bredin, "Tristounet: triplet loss for speaker turn embedding," in *ICASSP*. IEEE, 2017, pp. 5430–5434.

[11] Zhongxin Bai and Xiao-Lei Zhang, "Speaker recognition based on deep learning: An overview," *Neural Networks*, 2021.

[12] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Proceedings of Odyssey: The Speaker and Language Recognition Workshop*, 2018, pp. 74–81.

[13] Na Li, Deyi Tuo, Dan Su, Zhifeng Li, Dong Yu, and A Tencent, "Deep discriminative embeddings for duration robust speaker verification.," in *INTERSPEECH*, 2018, pp. 2262–2266.

[14] Yi Liu, Liang He, and Jia Liu, "Large margin softmax loss for speaker verification," *arXiv preprint arXiv:1904.03479*, 2019.

[15] Lantian Li, Zhiyuan Tang, Ying Shi, and Dong Wang, "Gaussian-constrained training for speaker verification," in *ICASSP*. IEEE, 2019, pp. 6036–6040.

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

[17] Yunqi Cai, Lantian Li, Andrew Abel, Xiaoyan Zhu, and Dong Wang, "Deep normalization for speaker vectors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 733–744, 2020.

[18] Mahdi Hajibabaei and Dengxin Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[19] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[20] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *APSIPA ASC*. IEEE, 2019, pp. 1652–1656.

[21] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "VoxCeleb2: Deep speaker recognition," in *INTERSPEECH*, 2018, pp. 1086–1090.

[23] Mitchell McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson, "The speakers in the wild (SITW) speaker recognition database.," in *INTERSPEECH*, 2016, pp. 818–822.

[24] Lantian Li, Ruiqi Liu, Jiawen Kang, Yue Fan, Hao Cui, Yunqi Cai, Ravichander Vipperla, Thomas Fang Zheng, and Dong Wang, "CN-Celeb: multi-genre speaker recognition," *arXiv preprint arXiv:2012.12468*, 2020.

[25] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "BUT system description to Voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.