A comparison of the binary cross-entropy and the soft detection loss function

Shreyas Ramoji, Sriram Ganapathy

February 17, 2022

Abstract

In this draft, we present an intuitive explanation to show how differently the gradient descent algorithm plays out for two choices of loss functions - The binary cross-entropy (BCE) and softDCF for binary classification task of speaker verification, using equations and plots. While both these functions penalize the model for making errors (false alarms and missed detection), only the BCE loss can cause a randomly initialized network to be capable of learning to discriminate between the target and non-target hypotheses. On the other hand, the softDCF loss is useful for fine-tuning pre-trained networks, due to its ability to correct errors close to the detection threshold.

1 The softDCF loss

In our paper on Neural PLDA [1], we proposed a loss function called softDCF based on the Bayes risk of a binary classification model, which is defined as:

$$C_B = \pi C_{Miss} P_{Miss}(\theta) + (1-\pi) C_{FA} P_{FalseAlarm}(\theta) \tag{1}$$

where π is the prior probability of hypothesis 1 (target hypothesis), C_{Miss} and C_{FA} are costs associated with the miss/false negative error respectively. P_{Miss} and P_{FA} are the probabilities of miss and false alarm errors respectively computed by applying a threshold θ to the output (score) of the system.

The detection cost function (DCF) is obtained by normalizing by $C_{Miss}\pi$:

$$DCF = C_{Norm}(\theta) = P_{Miss}(\theta) + \beta P_{FA}(\theta) \tag{2}$$

where $\beta = \frac{C_{FA}(1-\pi)}{C_{Miss}\pi}$.

The probabilities $P_{miss}(\theta)$ and $P_{FA}(\theta)$ are defined as

$$P_{Miss}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathbb{1}(s_i < \theta) \tag{3}$$

$$P_{FA}(\theta) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \mathbb{1}(s_i \geq \theta) \tag{4}$$

where $\mathcal{T}$ is the set of indices of class 1 (target), $\mathcal{N}$ is the set of indices of class 0 (non-target), s_i is the system output (score) for the i^{th} input.

The input to a speaker verification system is a pair of speech utterances. The target hypothesis is that the pair of utterances are spoken by the same speaker, and the non-target hypothesis is that the pair of utterances is from a different speaker. The most widely used evaluation metric for a speaker verification system is called minDCF or C_{min}, which the minimum C_{Norm} that an be achieved at an optimal threshold.

$$\text{minDCF} = C_{min} = \min_{\theta} C_{Norm}(\theta) \tag{5}$$

We would like to optimize a speaker verification model for the minDCF, but, the DCF contains step non-linearities at the threshold θ as can be seen in equations (3) and (4). Hence, we proposed a differentiable approximation of the DCF, called softDCF, by approximating the step non-linearities with sigmoid activations as follows:

$$\text{softDCF}(\beta, \theta) = P_{Miss}^{soft}(\theta) + \beta P_{FA}^{soft}(\theta) \tag{6}$$

where

$$P_{Miss}^{(\text{soft})}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} [1 - \sigma(\alpha(s_i - \theta))] \tag{7}$$

$$P_{FA}^{(\text{soft})}(\theta) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \sigma(\alpha(s_i - \theta)) \tag{8}$$

Here, $\mathcal{T}$ is the set of indices of the target trials, and $\mathcal{N}$ is the set of indices of the non-target trials in the training batch, and s_i is the output of the network (score) corresponding to the i^{th} training example (trial).

2 Comparing the two loss functions

We express the two loss functions of interest in a comparable format, and comment on the nature of these loss functions.

The softDCF loss can be written as:

$$softDCF = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} [1 - \sigma(\alpha(s_i - \theta))] + \frac{\beta}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \sigma(\alpha(s_i - \theta)) \tag{9}$$

The binary cross-entroy (BCE) loss function is defined as:

$$L_{BCE} = -\sum_i (t_i \log(\sigma(s_i)) + (1 - t_i) \log(1 - \sigma(s_i))) \tag{10}$$

which can be re-written as

$$L_{BCE} = -\sum_{i \in \mathcal{T}} \log(\sigma(s_i - \theta)) - \sum_{i \in \mathcal{N}} \log(1 - \sigma(s_i - \theta)) \tag{11}$$

Here, s_i is the log-likelihood ratio score, θ is the decision threshold, α is the sigmoid warping factor, $\mathcal{T}$ denotes the set of target trials and $\mathcal{N}$ denotes the set of non-target trials.

Now, equations (2) and (3) are in the same form, involving sums over different functions of the target and non-target scores output by the system.

We express the overall loss function as sum of loss functions on each sample,

$$L = \sum_{i \in \mathcal{T}} F(s_i) + \sum_{i \in \mathcal{T}} G(s_i) \tag{12}$$

For the softDCF loss,

$$F(s_i) = \frac{1}{|\mathcal{T}|} [1 - \sigma(\alpha(s_i - \theta))] \qquad G(s_i) = \frac{\beta}{|\mathcal{N}|} \sigma(\alpha(s_i - \theta)) \tag{13}$$

and for the BCE loss:

$$F(s_i) = -\log(\sigma(s_i - \theta)) \qquad G(s_i) = -\log(1 - \sigma(s_i - \theta)) \tag{14}$$

Differentiating equation (4) with respect to the network parameters Λ, we get

$$\frac{\partial L}{\partial \Lambda} = \sum_{i \in \mathcal{T}} \frac{\partial F(s_i)}{\partial s_i} \frac{\partial s_i}{\partial \Lambda} + \sum_{i \in \mathcal{N}} \frac{\partial G(s_i)}{\partial s_i} \frac{\partial s_i}{\partial \Lambda} \tag{15}$$

Denoting $\frac{\partial F(s_i)}{\partial s_i}$ as $f(s_i)$ and $\frac{\partial G(s_i)}{\partial s_i}$ as $g(s_i)$ and rewriting the equation, we get:

$$\frac{\partial L}{\partial \Lambda} = \sum_{i \in \mathcal{T}} f(s_i) \frac{\partial s_i}{\partial \Lambda} + \sum_{i \in \mathcal{N}} g(s_i) \frac{\partial s_i}{\partial \Lambda} \tag{16}$$

Equation (8) is a weighted sum of the gradient of the network outputs w.r.t the parameters. These weights are a function of the output s_i, and the ground truth. The "gradient descent" update is given by

$$\Lambda^{(new)} \longleftarrow \Lambda^{(old)} - \eta \frac{\partial L}{\partial \Lambda} \tag{17}$$

where η is a sufficiently small enough positive learning rate. Substituting for $\frac{\partial L}{\partial \Lambda}$ from eqn (16) in eqn (17), we can write

$$\Lambda^{(new)} \longleftarrow \Lambda^{(old)} - \left(\sum_{i \in \mathcal{T}} \eta f(s_i) \frac{\partial s_i}{\partial \Lambda} + \sum_{i \in \mathcal{N}} \eta g(s_i) \frac{\partial s_i}{\partial \Lambda} \right) \tag{18}$$

If the gradient descent algorithm has to cause the target scores to increase above a threshold and the non-target scores to decrease below the threshold, $f(s_i)$ has to be non-positive and $g(s_i)$ should be non-negative. By examining how $f(s_i)$ and $g(s_i)$ vary with s_i, for both the loss functions, we can comment on how differently they are from each other in achieving the objective of separating the target and non-target distributions.

For the BCE loss (Repeating eqn (14)), we have:

$$F(s_i) = -\log(\sigma(s_i - \theta)) \qquad G(s_i) = -\log(1 - \sigma(s_i - \theta)) \tag{19}$$

$$f(s_i) = -\left[1 - \sigma(s_i - \theta)\right] \qquad g(s_i) = \sigma(s_i - \theta) \tag{20}$$

The plot of $f(s_i)$ and $g(s_i)$ are shown in Figure 1 and Figure 2.

The BCE loss function satisfies the above mentioned criterion that $f(s_i)$ should be non-positive and $g(s_i)$ should be non-negative. $f(s_i)$ has a value of -0.5 at the threshold, and quickly converges to -1 asymptotically as s_i reduces below the threshold, and quickly converges to 0 asymptotically as s_i increases above the threshold. Similarly, $g(s_i)$ has a value of 0.5 at the threshold, and quickly converges to +1 asymptotically as s_i increases above the threshold, and quickly converges to 0 asymptotically as s_i decreases below the threshold.

On the contrary, in the case of the softDCF loss (eqn (1)), we have

$$F(s_i) = \frac{1}{|\mathcal{T}|} \left[1 - \sigma(\alpha(s_i - \theta))\right] \qquad G(s_i) = \frac{\beta}{|\mathcal{N}|} \sigma(\alpha(s_i - \theta)) \tag{21}$$

$$f(s_i) = -\frac{\alpha}{|\mathcal{T}|} \sigma(\alpha(s_i - \theta)) \left[1 - \sigma(\alpha(s_i - \theta))\right] \qquad g(s_i) = \frac{\alpha\beta}{|\mathcal{N}|} \sigma(\alpha(s_i - \theta)) \left[1 - \sigma(\alpha(s_i - \theta))\right] \tag{22}$$

The functions $f(s_i)$ and $g(s_i)$ are bell shaped curves around the threshold, whose width and height are controlled by the warping factor α. This means only the examples whose scores are within the support of the bell shaped curves (between 2 and 4 in the illustrated plot) are the ones that contribute to the gradient update.

To summarize, the gradient update rule using BCE loss updates the parameters in a direction that causes all the scores of the target training examples below the threshold to increase, and all the scores of the non-target training examples above the threshold to decrease. This happens repeatedly, batch after batch and epoch after epoch, eventually causing almost all the target and non-target score distributions to distinctively separate out at the threshold. On the other hand, the gradient update

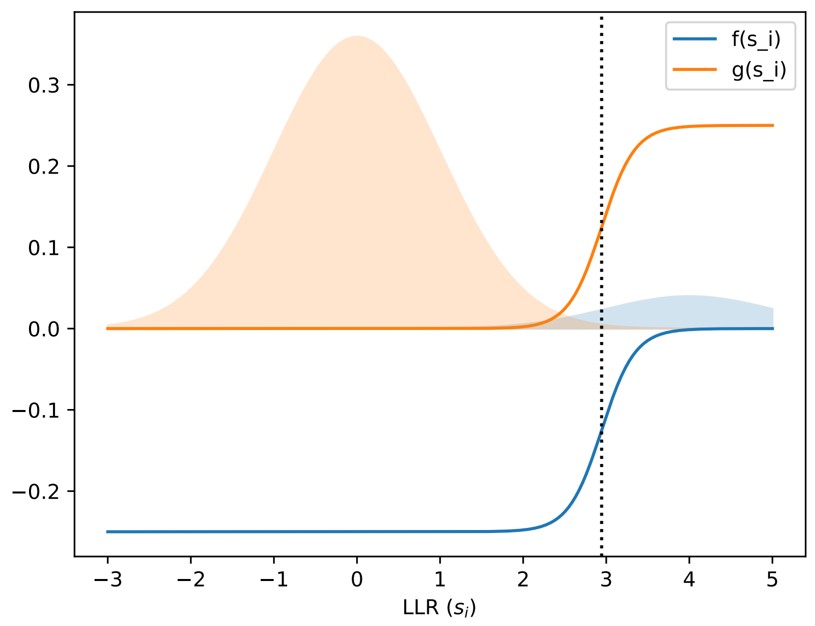

Figure 1: The plots of $f(s_i)$ and $g(s_i)$ for BCE loss. The vertical dotted black line is used to denote the threshold θ. For illustration purpose, we show the non-target score distribution in orange(on the left) and the target score distribution in blue (on the right).

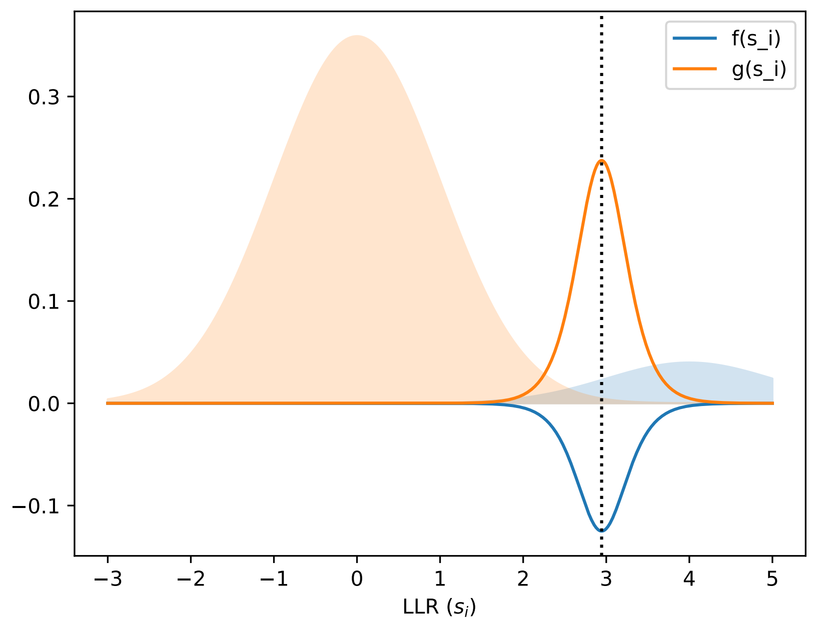

Figure 2: The plots of $f(s_i)$ and $g(s_i)$ for softDCF loss. The vertical dotted black line is used to denote the threshold θ. For illustration purpose, we show the non-target score distribution in orange(on the left) and the target score distribution in blue (on the right).

rule using softDCF loss updates the parameters in a direction such that the target scores in a small neighbourhood around the threshold are encouraged to increase, and the non-target scores in the same region to decrease, thereby correcting the errors in a small neighbourhood around the threshold.

When the network is randomly initialized, the target and non-target score distributions are highly overlapping, and also have larger gradients compared to a pre-trained network. In such cases, as the P_{FA} term has a higher weight (by a factor of β), all the scores (target and non-target) are found to quickly move to the left side of the bell curve where $P_{FA}^{(soft)} \approx 0$ and $P_{miss}^{(soft)} \approx 1$. This is a saddle point of the softDCF loss, where the gradients are always close to zero, and does not ensure that the target and non-target distributions get separated. Thus, most of the samples end up outside the support of the bell shaped $f(s_i)$ or $g(s_i)$ curves. This is a saddle point of the softDCF loss function where the gradient is always close to zero, and does not ensure that the target and non-target distributions get separated.

One potential direction to overcome this issue would be to consider the BCE loss for the initial training followed by the softDCF loss for the final training. This is the equivalent to the pre-trained initialization performed in the proposed approach, where the model is pre-trained in order to have the target and non-target scores partially separated before the training of the model with softDCF loss.

References

[1] Shreyas Ramoji, Prashant Krishnan, and Sriram Ganapathy. NPLDA: A Deep Neural PLDA Model for Speaker Verification. In *Odyssey*, pages 202–209, 2020.

This is a supplementary write-up to the journal draft "PLDA inspired Siamese Networks for Speaker Verification" submitted to Computer Speech and Language.