Experiments report

PS：红色表示需要修改的地方

Introduction

The goal of this work is to improve SRE performance in PLDA score using Variational Autoencoder and triplet loss.

Probabilistic linear discriminant analysis (PLDA) is the defector standard for backends in i-vector speaker recognition. But if we try to extend the PLDA in d-vector or x-vector SRE system, the performance is not as well as i-vector. One of the main causes of this problem is that i-vector follows Gaussian distribution, but d-vector and x-vector do not.

Some predecessors (Lilt...) think that there are 2 main factors determine the PLDA performance:

1. **the Gauss property of embedding**
2. **Distinguishability between different speaker's embedding**

In this work, we propose to approach this problem using stochastic gradient variational Bayes, which aims to enhance the Gauss property of d/x-vector. And we also add triplet loss into model loss function to increase the distinguishability between different speaker's embedding.

用中文重新表述一下·

决定 vector 在 PLDA 的好坏有两个因素，

1. vector 是否有很强的高斯性
2. 不同说话人之间的 vector 是否有很强的区分性

这个实验的目标：

用 VAE 使 vector 高斯+降维（但可能会降低区分性）
因此用 triplet 使得 vector 保证有区分性

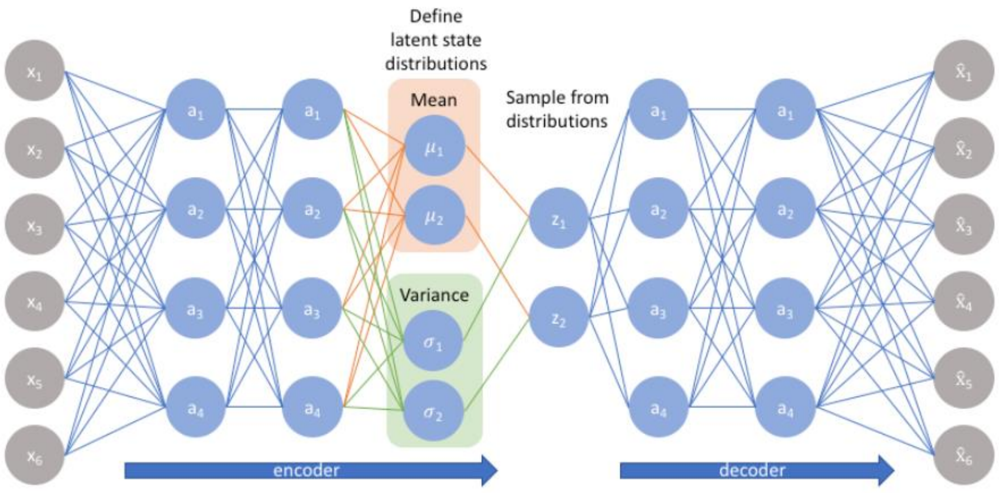
总结来说，用 VAE 使 vector 变得高斯的情况下用 triplet loss 保证区分性，从而提高 PLDA 的性能

Experiments

Network structure

1. Vanilla VAE

VAE encoder: 2 hidden layers (sigmoid+tanh)
VAE decoder: 2 hidden layers (tanh+sigmoid)


Define latent state distributions
Mean
Variance
Sample from distributions
encoder
decoder
x_1 x_2 x_3 x_4 x_5 x_6
a_1 a_2 a_3 a_4
μ_1 μ_2
σ_1 σ_2
z_1 z_2
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$

We trained this model using **reparameterization trick**, this enables us to optimize of the loss function using backpropagation to jointly estimate the parameters that define the model and the approximate posteriors of the latent factors

$$\mathcal{L}(x, \hat{x}) + \sum_j KL\left(q_j\left(z|x\right)||p\left(z\right)\right)$$

2. VAE with Triplet Loss

结构图略

$$\underline{\text{Total\_loss} = \text{Loss (VAE)} + \alpha \text{ loss (triplet loss)}}$$

Work Implementation

1. Firstly, we use kaldi toolkit to extract d-vector and x-vector with 512 dimensions from utterance.
2. Then, we implemented the VAE with the TensorFlow toolkit, the inputs are d-vector and x-vector extracted from the first step.
3. When we finish the train training process, we input the original d/x-vector into network and get the output from VAE encoder.
4. At last, we use kaldi toolkit to calculate EER of output vector.

NOTE: I have submitted the TF-VAE code to GitHub. (https://github.com/zyzisyz/zkhhk)

Datasets

Train: voxceleb_combined_200000
Test: sitw_dev & sitw_eval

Baseline

	Skew	Kurt	Cosine EER	PLDA EER	LDA PLDA
d-vector	0.1092	-0.11609	38.39%	17.71%	9.511%
x-vector	-0.04228	-0.3603	15.67%	9.087%	3.157%
d-vector-LDA	-0.005515	0.028395	12.94%	9.665%	9.434%
x-vector-LDA	-0.01074	-0.0089	5.198%	3.735%	3.157%

Experiment Results:

We designed 4 groups of experiments to find the best network parameters.

1. D/X-vector -> Vanilla VAE
2. D/X-vector -> VAE with Triplet Loss
3. D/X-vector -> LDA -> Vanilla VAE
4. D/X-vector -> LDA ->VAE with Triplet Loss

以下实验结果为 KL 和 MSE 等权重

1. D/X-vector -> Vanilla VAE

Vanilla VAE Best Result

	The Best Network structure	Cosine EER		PLDA EER		LDA PLDA EER	
		baseline	VAE	baseline	VAE	baseline	VAE
d-vector	Z dim: 600 layer unit: 1800	38.39%	**38.54%**	17.71%	**16.02%**	9.511%	**12.59% (75)**
x-vector	Z dim: 200 layer unit: 2300	15.67%	**13.79%**	9.087%	**5.006%**	3.157%	**4.236% (100)**

The PLDA EER of d-vector does not have an obvious enhancement. But the X-vector's result seems good.

Note:

1. Baseline means without VAE encoder.
2. The best d-vector-VAE's Z dim is 600 which is more than 512.
3. LDA PLDA EER 括号表示 LDA 的 dim

2. D/X-vector -> VAE with Triplet Loss

The total loss function of VAE with Triplet Loss consists of 2 parts (VAE Loss + Triplet Loss), in this experiment, we use the Vanilla VAE best network structure we have got before to find the best hyperparameter α.

$$\underline{\text{Total\_loss} = \text{Loss (VAE)} + \alpha \text{ loss (triplet loss)}}$$

	The Best Network structure	Cosine EER		PLDA EER		LDA PLDA EER	
		Baseline	VAE	baseline	VAE	baseline	VAE
d-vector	α: 150	38.39%	**16.1%**	17.71%	**13.67%**	9.511%	**11.17% (65)**
x-vector	α: 200	15.67%	**6.546 %**	9.087%	**4.197%**	3.157%	**4.043% (130)**

3. D/X-vector -> LDA -> Vanilla VAE

	The Best Network structure	Cosine EER		PLDA EER		LDA PLDA EER	
		baseline	VAE	baseline	VAE	baseline	VAE
d-vector	Z dim: 80 layer unit: 1600	12.94%	**12.09%**	9.665%	**9.472%**	9.434%	**9.049%（70）**
x-vector	Z dim: 120 layer unit: 1800	5.198%	**4.39%**	3.735%	**3.581%**	3.157%	**3.273%（110）**

4. D/X-vector -> LDA ->VAE with Triplet Loss

	The Best Network structure	Cosine EER		PLDA EER		LDA PLDA EER	
		baseline	VAE	baseline	VAE	baseline	VAE
d-vector	α=30	12.94%	**10.74%**	9.665%	**9.472%**	9.434%	**9.203%(85)**
x-vector	α=6	5.198%	**4.736%**	3.735%	**3.926%**	3.157%	**3.735%（80）**

最终，我和蓝天学长选择的网络结构为

512(输入)->1800->1800->200VAE(输出)

512(输入)->LDA->150-> 1600->1600->100(输出)

总的来说，以上实验结果表明，

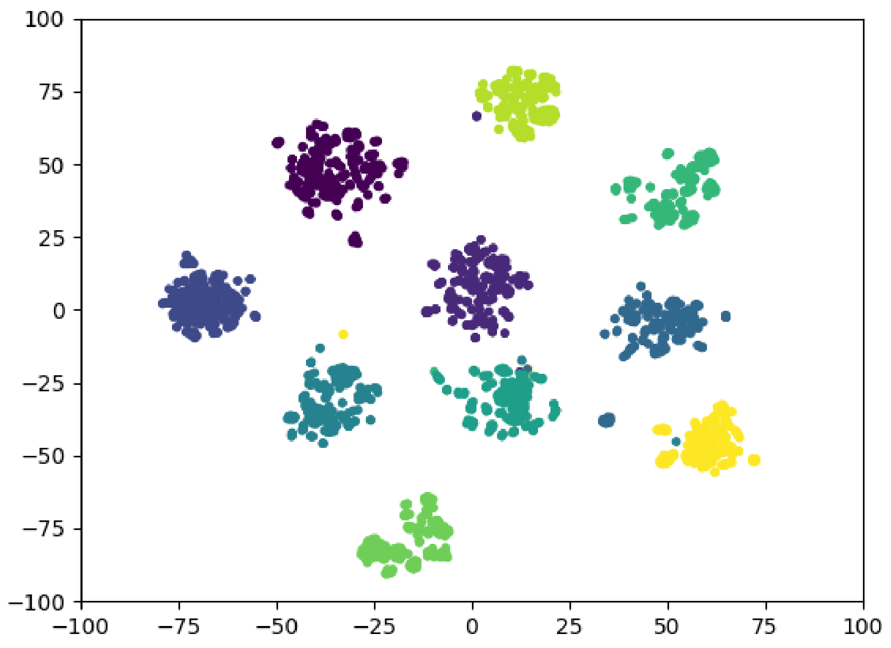
1. **VAE 在 x-vector 的 PLDA EER 上有着非常明显的作用**
2. **Triplet Loss 在 Cosine EER 有着非常明显的作用**
3. **vector 本身的区分性已经很强了，Triplet Loss 的 Loss 值很小，小于 1，因此所乘的权重很高，我设置的为 100。**

X-vector 的 Tsne 图

X

Skew: -0.04228, Kurt: -0.3603

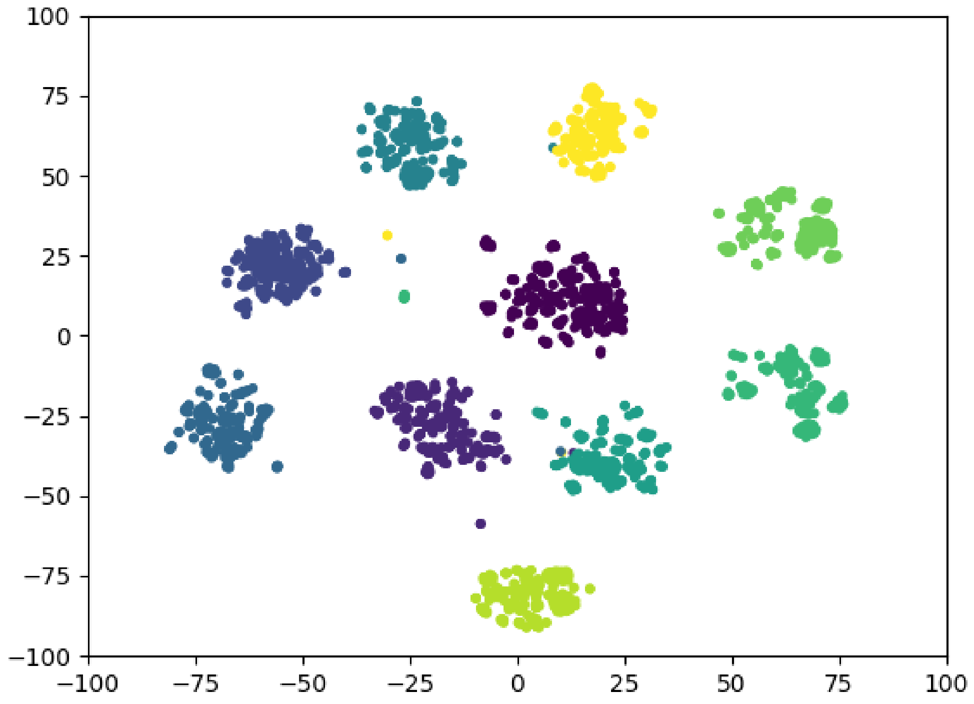
Cosine EER: 15.67%: , PLDA EER: 9.87%



X-VAE

Skew: -0.00436, Kurt: 0.0346

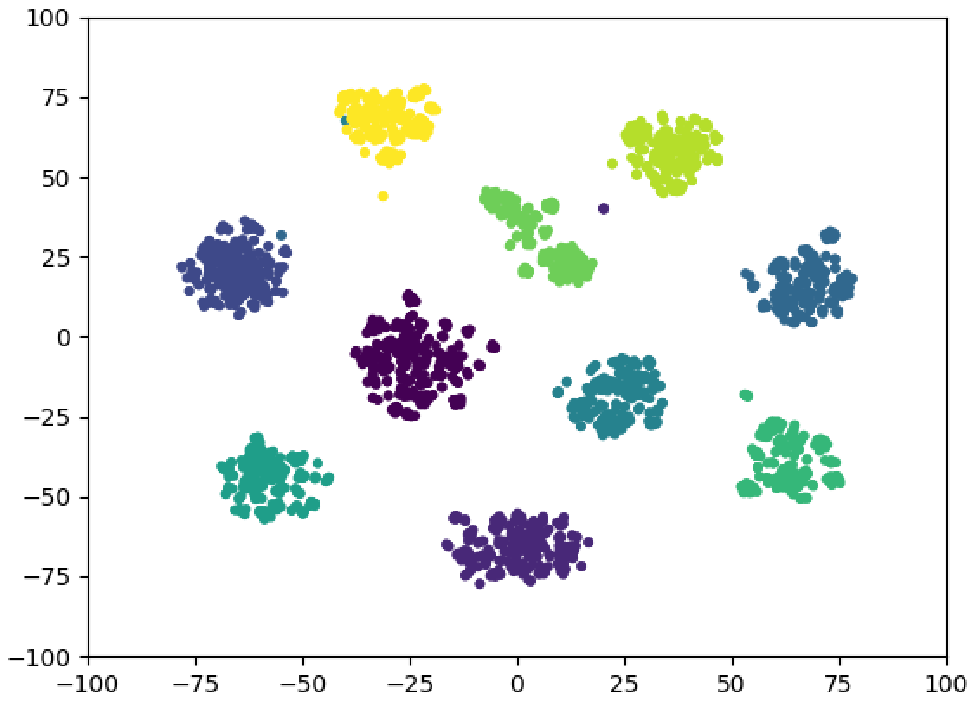
Cosine EER: 13.76%, PLDA EER: 5.006%



X-VAE_TL

Skew: -0.01047, Kurt: -0.03296

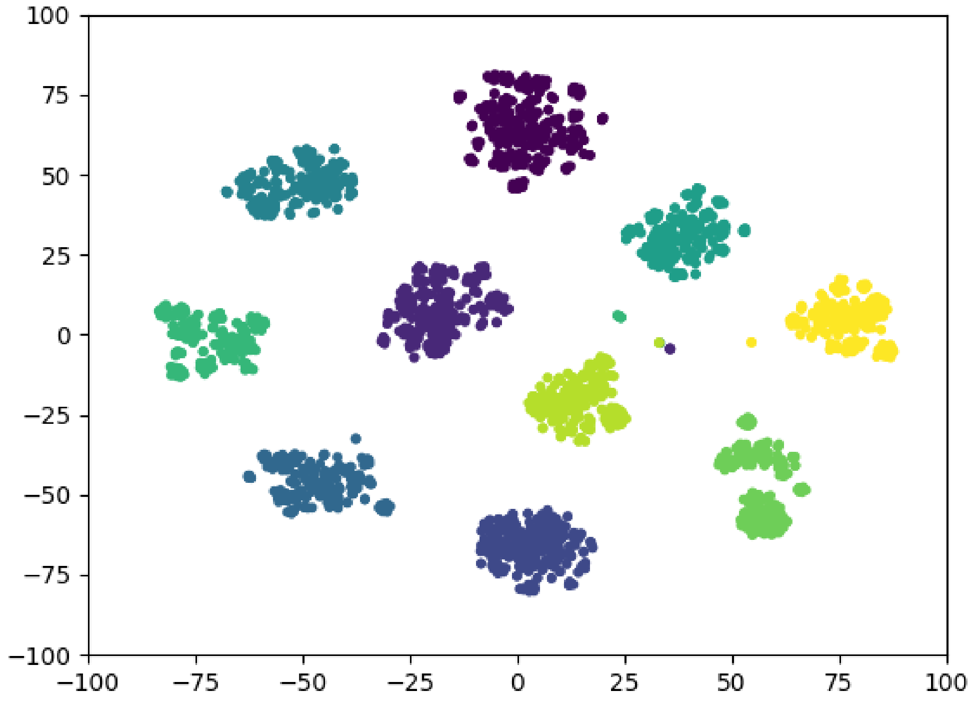
Cosine EER: 6.252% PLDA EER: 4.179%



X-LDA

Skew: -0.0107, Kurt: -0.00894

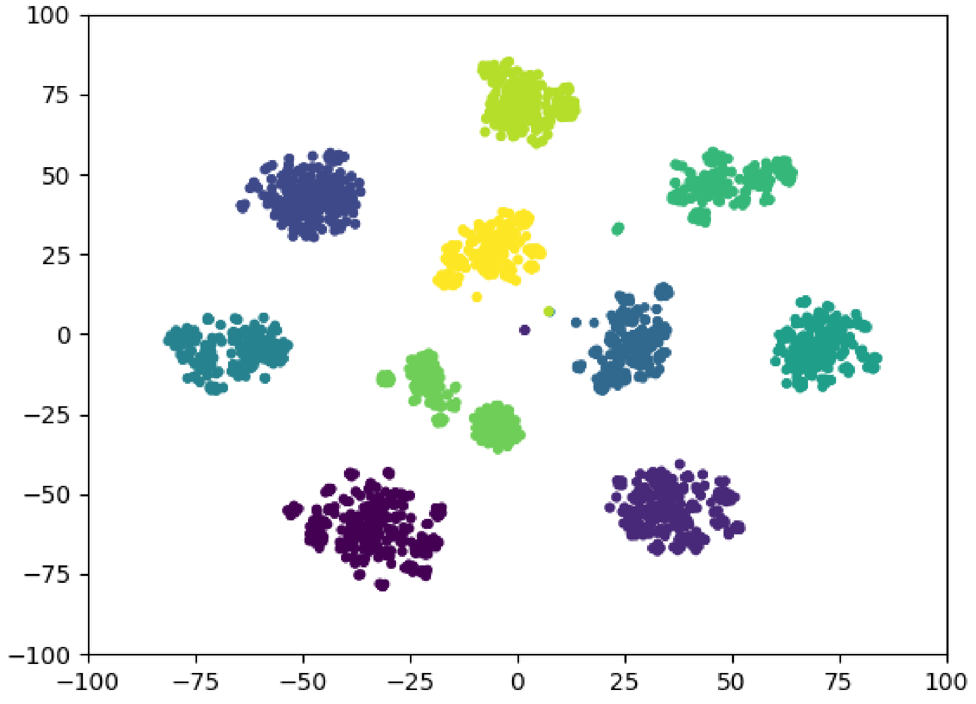
Cosine EER: 5.198% PLDA EER: 3.735%



X-LDA-VAE

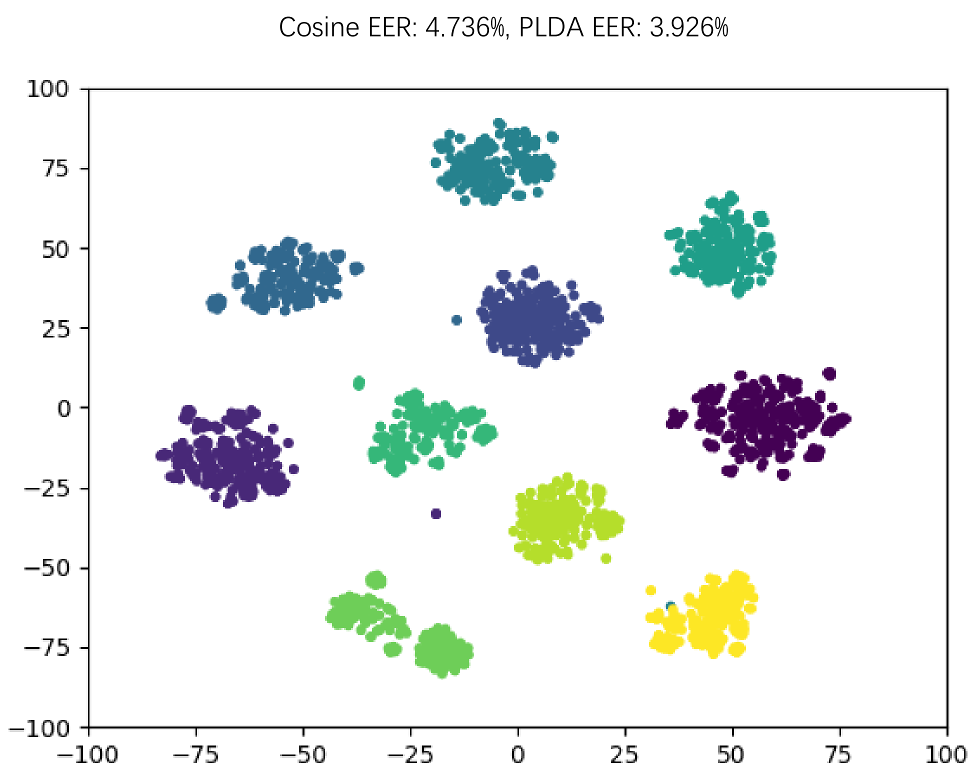
skew: 0.00277, Kurt: -0.0378

Cosine EER: 4.39% , PLDA EER: 3.581%



X-LDA-VAE_TL

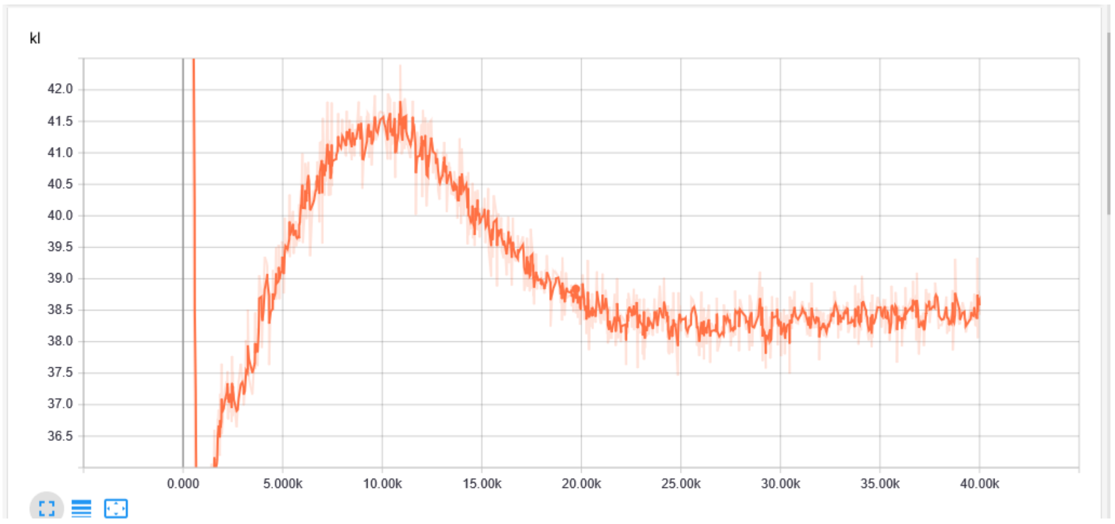
Skew: 0.00291, Kurt: -0.0536



Cosine EER: 4.736%, PLDA EER: 3.926%
100
75
50
25
0
−25
−50
−75
−100
−100
−75
−50
−25
0
25
50
75
100

但实验中我们发现,

1. KL loss 收敛的很奇怪

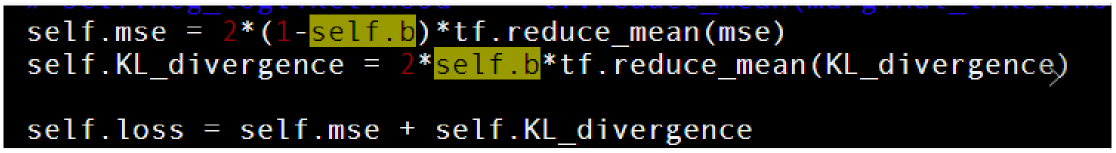


2. 因此修改了 Vanilla VAE 中 KL 所占的权重，实验发现，当 KL 权重很低时，即 MSE 所占权重很高时，PLDA EER 性能更好。

3. Triplet loss 可以把 Cosine EER 降得很低，但 PLDA EER 反而没有纯 VAE 的好

以下实验结果为

寻找 KL 和 MSE 最优权重所做实验



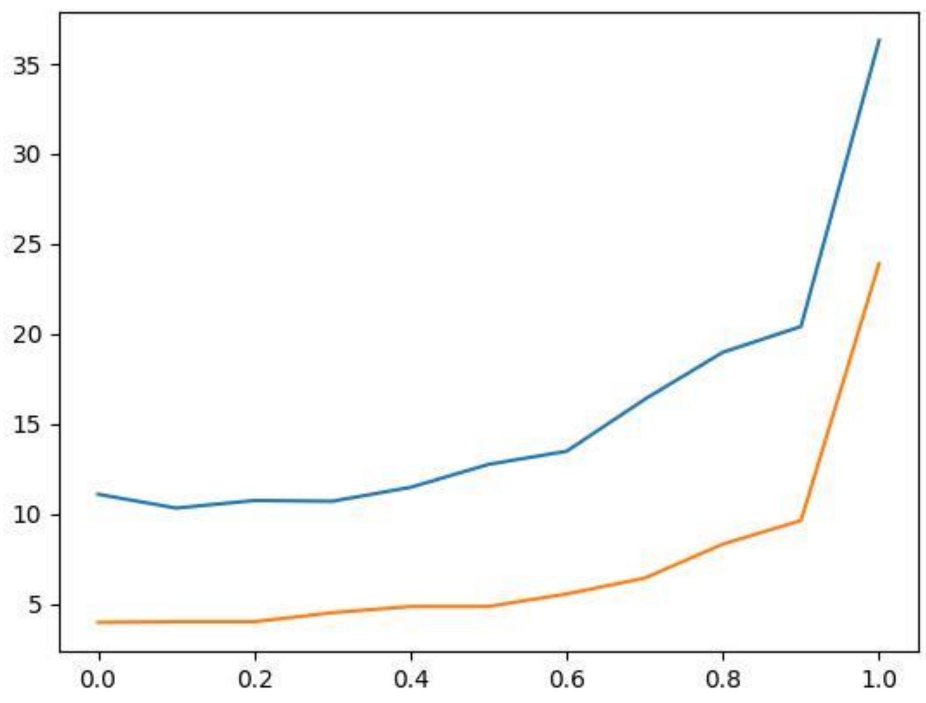
```
self.mse = 2*(1-self.b)*tf.reduce_mean(mse)
self.KL_divergence = 2*self.b*tf.reduce_mean(KL_divergence)

self.loss = self.mse + self.KL_divergence
```

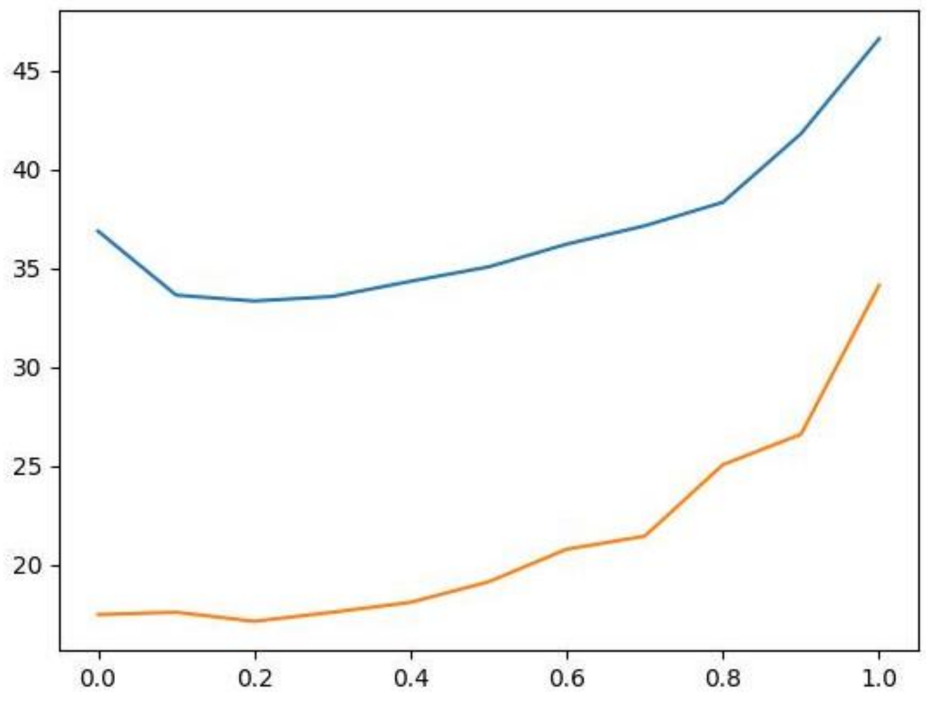
横坐标是 KL 所占权重，纵坐标是 PLDA EER 和 Cosine EER

当横坐标是 0.5，KL 和 MSE 是等权重

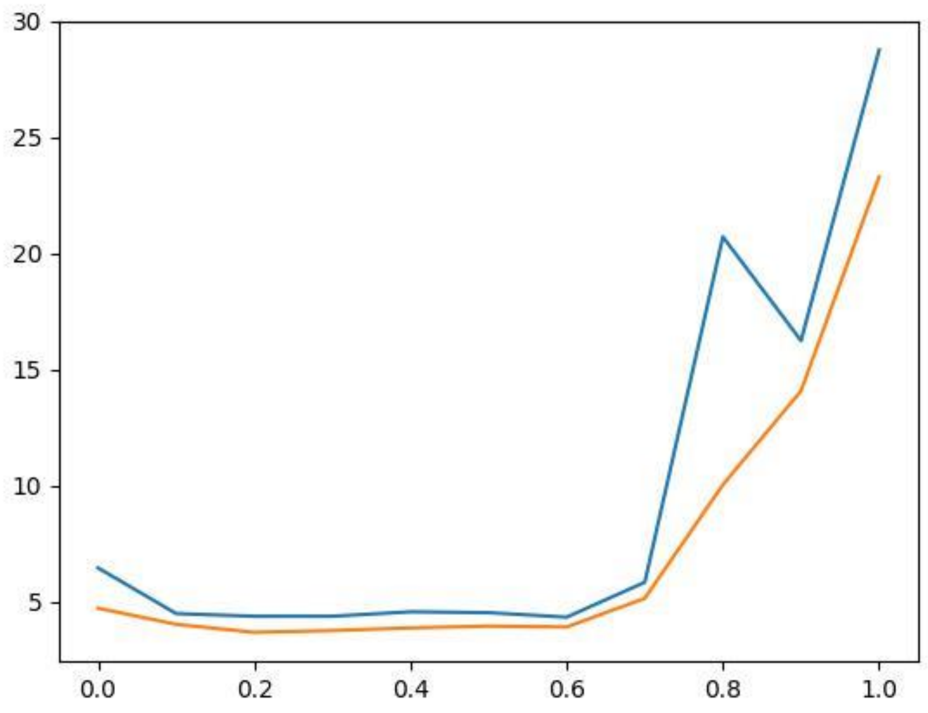
X-vector 训练了 40 个 epoch 每个 epoch 1000 个 batch



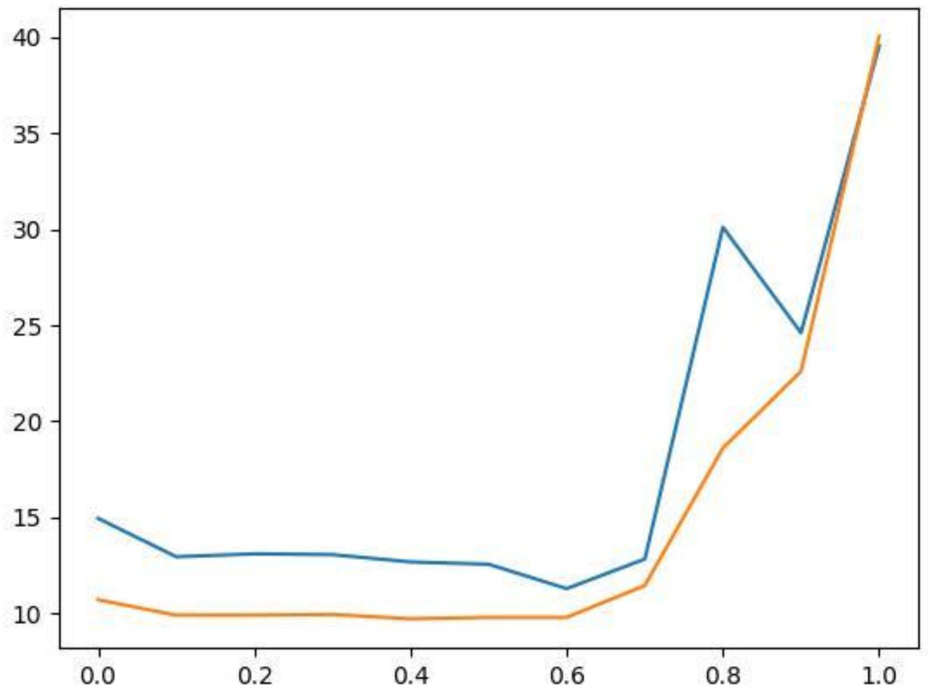
D-vector 训练了 40 个 epoch 每个 epoch 1000 个 batch



X-vector 经过 LDA 后训练了 40 个 epoch 每个 epoch 1000 个 batch



D-vector 经过 LDA 后训练了 40 个 epoch 每个 epoch 1000 个 batch

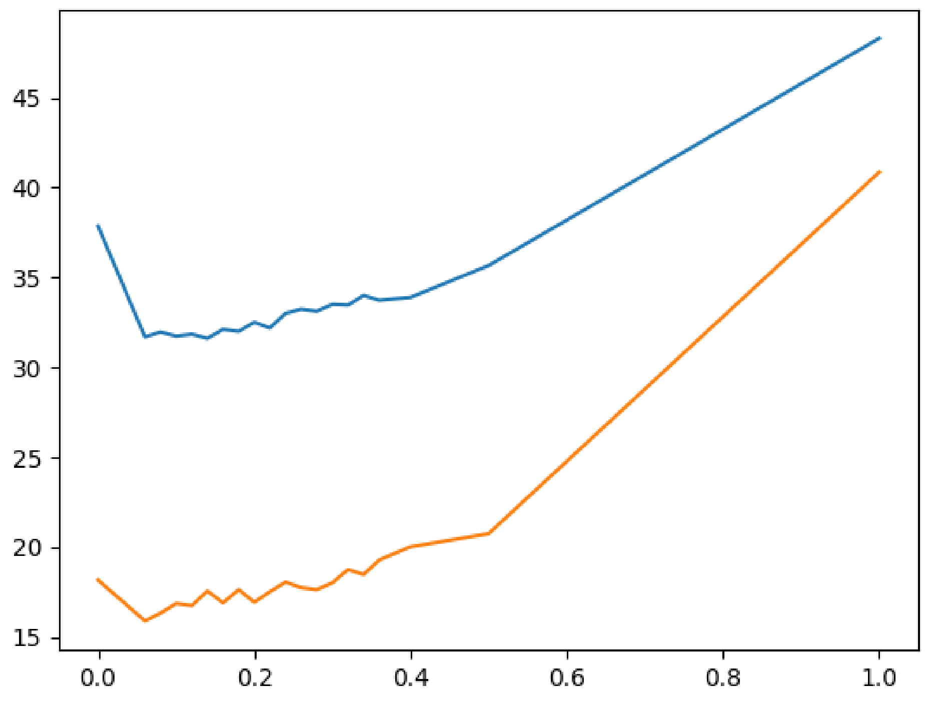


从上面四张图，我们发现 KL 的权重应该小于 0.5
因此我在 0.06 与 0.5 之间做了更加精细的实验

注：实验结果显示 KL 散度有用，但权重不能太高，得非常低

感觉 Loss 还没完全收敛，我还需要再测试一下

但大致的图为



现在我和蓝天哥的设想是
1. 调节 KL 与 MSE 还有 Triplet Loss 的权重
2. 采用 PCA 做一组实验
3. 或者修改 EER 的计算方法，直接使用 predict 出来的 mean 和σ计算 KL 距离