Speaker separation and diarization with deep learning

Eric Bolo

Batvoice

February 22, 2018

Motivation

State of the art prior to deep learning:

- Non-negative Matrix Factorization (NMF)
- CASA: Computational Auditory Scene Analysis
  - Generates embeddings with specialized, hand-picked features

Problems:

- hard to optimize
- performs poorly with low quality audio, e.g. 8KHz, 8-bit, mono telephone recordings from PVCP.

Deep clustering (1)

Deep learning has been successful in speaker-dependent separation and for many speech enhancement tasks, but until recently underperformed when given the task of separating unknown speakers.

Deep learning methods for separation generally work by applying a mask to the mixture spectrogram, one mask per speaker.

Without prior knowledge of the speakers, the network must preserve the order of the speakers in the outputs, i.e. preserve the same output mask for the same speaker across time. It's the "permutation problem" [1].

Deep clustering (2)

To tackle permutation, Isik et al. frame the problem as one of clustering rather than classification.

The goal is to cluster together input TF-bins where the same source dominates, without explicitly assigning the bins to a source [1].

Deep clustering (3)

Raw input signal: x

Time-Frequency spectrogram index: (t, f)

Feature vector: $X_i = g_i(x)$, $i \in 1, .., N$, $X_i = X_{(t,f)}$, g: STFFT

Target partition (ground-truth): $Y = \{y_{i,c}\}$

$$y_{i,c} = \begin{cases} 1 & \text{if } c \text{ is the dominant source at } i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Deep clustering (4)

$A = YY^T$
$A \in R^{NxN}$ is a binary affinity matrix such that:

$$A_{i,j} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same cluster} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

YY^T is permutation-invariant, i.e. indifferent to the ordering of the clusters c:

$$(YP)(YP)^T = YY^T \text{for any permutation matrix } P \quad (3)$$

Deep clustering (5)

The model produces an approximation $\hat{A}$ of A by generating embeddings, i.e. high-dimensional representations of T-F bins that are suitable for clustering.

$$V = f_\theta(X) \in R^{NxD} \tag{4}$$

Where:

- θ: parameters of the model
- D: chosen embedding dimension
- $|v_i|^2 = 1$

Deep clustering (6)

Cost function:

$$C_Y(V) = ||\hat{A} - A||_F^2 = ||VV^T - YY||_F^2 \quad (5)$$

During inference, the embeddings V are clustered using K-means.

Network architecture: 4 BLSTM layers + 1 fully connected forward layer with tanh activation

Deep clustering (7)

Deep clustering outperforms baseline models (NMF, CASA) by a large margin.

But... the model only handles the permutation problem at the utterance level. Memory limitations impose a maximum input length causing the input audio file to be chunked into utterances whose outputs are independent. A post-clustering step is needed to connect separated sources across utterances. Hence:

- no end-to-end optimization
- no real-time processing

Deep Attractor Networks (1)

DANet starts from the same structure as DPCL: a stack of BLSTM layers topped by a fully-connected layer which generates D-dimensional embeddings for all TF-bins:

$$V = f_\theta(X) \in R^{NxD} \tag{6}$$

(Borrowing notation from the DPCL paper [1] for consistency)

The difference lies in how the model computes the loss. In DPCL, the loss is the Frobenius norm of the distance between an ideal and an estimated affinity matrix. DANet instead uses "attractors" in the embedding space to compute similarity [2]

Deep Attractor Networks (2)

An attractor $A \in R^{1xD}$ represents a specific speaker.
During training, the attractors represent the centroid of each speaker:

$$A_i = \frac{Y_i V}{\sum_{f,t} Y_i}, i \in [1, C] \quad (7)$$

Where:

- $Y_i \in R^{1xN}$ contains the speaker assignments for speaker i, for each TF-bin (binary or real values between 0 and 1).

Deep Attractor Networks (3)

Power filter: robustify model by taking into account only those TF-bins where power is greater than some threshold ρ:

$$W = \begin{cases} 1 & \text{if } X > \rho \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$A_i = \frac{(Y_i \odot W)V}{\sum_{f,t}(Y_i \odot W)}, i \in [1, C] \tag{9}$$

Where $\odot$ is element-wise multiplication.

Deep Attractor Networks (4)

Define $D_i \in R^{1xN}$ to be the distance of each TF-bin in the embedding space from the attractor i.

$$D_i = A_i V^T, i \in [1, C] \tag{10}$$

Estimated mask for each speaker:

$$\hat{M}_i = H(D_i), i \in [1, C] \tag{11}$$

Where H is a non-linear function (sigmoid or softmax) which normalizes the distances s.t. $0 < H(D_i) < 1, \forall D_i$.

Deep Attractor Networks (5)

Given ground truth masks M_i, the loss is the L_2 reconstruction error:

$$L = \frac{1}{C}|X \odot (M_i - \hat{M}_i)|_2^2 \tag{12}$$

The masks are a function of both the attractors and embeddings, so the loss incentivizes the network to pull same-speaker embeddings together and pull the attractors away from each other.

Deep Attractor Networks (6)

How to estimate attractors during inference?

- Cluster the embeddings with K-means and define the attractors to be the centers of the clusters
- Define fixed attractors: take the mean of the attractors from all training mixtures, and use those averages as inference attractors
- Anchored DANet (ADANet): Teach the model to generate attractors during training and inference using "anchors" in the embedding space

Deep Attractor Networks (7)

ADANet uses trainable anchors, vectors in the embedding space, to estimate speaker assignment Y when both training and inferring. Y is then used to find the attractors as in the original DANet.

ADANet initializes C_{max} anchors:

$$B_j \in R^{1xD}, j \in [1, C_{max}] \tag{13}$$

Where C_{max} is the maximum number of speakers in the training mixtures.

Deep Attractor Networks (8)

In a mixture with C speakers:

- select all C combinations of the C_{max} anchors, denoted by L_p
  - $L_p \in R^{CxD}, p \in [1, \binom{C_{max}}{C}]$
- find the distance of the embeddings from the anchors in each subset
  - $D_p = L_p V^T, p \in [1, \binom{C_{max}}{C}]$
- use the distances to estimate speaker assignment:
  - $\hat{Y}_p = Softmax(D_p)$

Deep Attractor Networks (9)

For each anchor subset p, C attractors are generated taking $\hat{Y}_p$ as the speaker assignments.

$$A_p \in R^{CxD}, p \in \binom{C_{max}}{C} \tag{14}$$

ADAnet selects the set of attractors which maximizes the distance between attractors within the set.

$$S_p = A_p A_p^T, S_p \in R^{CxC} \tag{15}$$

$$s_p = \max S_{p_{ij}}, i \neq j \tag{16}$$

$$\hat{A} = \arg\min_{A_p} s_p, pp_i n \binom{C_{max}}{C} \tag{17}$$

Deep Attractor Networks (10)

ADANet pros

- no need for speaker assignment during inference
- fully end-to-end

ADANet cons

- increases computational complexity
- "since the correct permutation of the anchors is unknown, permutation invariant training is required for training the ADANet" (???)

[1] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R. Hershey.
Single-channel multi-speaker separation using deep clustering, 2016.

[2] Yi Luo, Zhuo Chen, and Nima Mesgarani.
Speaker-independent speech separation with deep attractor network, 2017.