Cheat sheet for "Deep clustering: Discriminative embeddings for segmentation and separation"

Bruce Sharpe, Feb. 14, 2017

Note: Terms introduced for the first time are put in *italics* and are explained in the Glossary at the end. The Glossary also includes terms that are in the paper, but not in the cheat sheet description.

What problem are they trying to solve?

Given a recording of two (or more) people speaking, split out the voices into separate recordings (*blind source separation*).

They suggest that their technique could also be useful for image segmentation, but that is not explored in this paper.

Why is this an interesting problem?

One important example of this is the *cocktail party* problem, and the authors seem to have made a significant advance on this old and difficult problem.

Progress in this area could improve voice control of robots, voice input for things like Amazon Echo, audio for mobile phones, and could lead to better hearing aids.

How has this been attempted previously?

People have worked on this for decades. Unfortunately for the clarity of the paper, some of that work is referenced in passing and can be a little confusing for those not familiar with it.

Suffice it to say that a whole lot of work has been done, but this new technique is pretty self-contained and you don't really need to know what happened before.

What's the general idea of the technique?

1. Convert the recording into a *spectrogram*
2. There will be regions in the spectrogram where one speaker dominates over the other, or no one is speaking
3. The goal is to train a neural network to automatically partition the spectrogram into those regions
4. Given a test recording, you run it through the trained network to partition its spectrogram. You use each partition segment as a binary mask on the spectrogram to extract the parts related to just one of the speakers at a time. You then do the reverse STFT to recover the sound of that speaker.
5. This last step, the recovery of individual speakers as listenable audio, is kind of glossed over in the paper. Subsequent papers expand on and improve this part of the procedure.

How is the partitioning done?

1. Each point in the spectrogram is converted to a D-dimensional vector (*embedding*)
2. The vectors are grouped into K clusters using standard techniques such as k-*means*. The number K is the number of speakers.
3. Deep clustering trains a neural network to learn the embedding.

What is the network and how is it trained?

The network is a *bidirectional recurrent network*. The *objective function* is a weighted squared distance that is designed to minimize the distances between embeddings of points within a partition, while maximizing the distances between embeddings of points in different partitions.

They generate training data from a standard corpus of recordings of individual speakers. They mix the recordings together (two speakers at a time) to create a training set where the correction partition is known.

Why is it called Deep Clustering?

There's a bit of marketing spin here. It's called "deep" because it uses a neural network. It's actually not very deep though, as neural networks go.

The "clustering" part almost makes sense because clustering is a key step in the process. However, I would quibble that it's not the clustering that is learned by the (not very) deep network, it is the embedding. I guess "deep embedding" brings the wrong connotations to mind.

How well does it work?

The authors use the best of the previous "traditional" techniques as a benchmark for how well the new technique works. It performs well as measured by a rather abstract metric (SDR) that is a little bit removed from the real problem we're trying to solve.

Subsequent papers take more of an end-to-end approach and measure their success by how well the output speech can be recognized by automatic speech recognition (ASR) algorithms. The results show a dramatic improvement over traditional approaches. The improvement is both quantitative and can be heard in their demo.

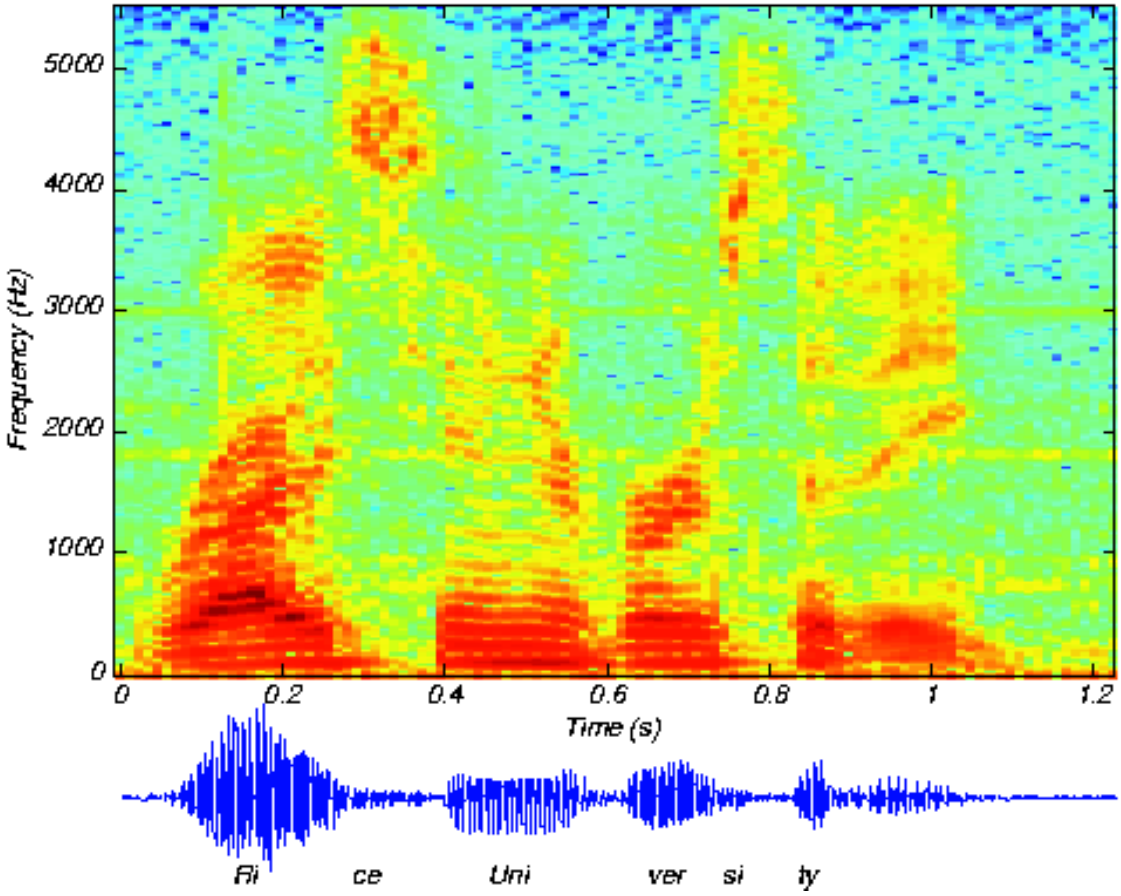
Glossary

- **affinity matrix:** Given n points and some concept of similarity between pairs of points, the affinity matrix is an $n \times n$ matrix where the entry (i, j) is the similarity between points i and j.
- **bidirectional recurrent network:** Neural networks are generally built up of two types: feed-forward and recurrent. I won't even attempt a description in this cheat sheet. Let's just say that they are a fancy way to map a complicated input (e.g., an image) into a simpler description (e.g., a descriptive label, like "dog").
- **blind source separation:** splitting an audio recording of mixed sounds into separate components, without explicitly knowing much about the sources, the recording setup, etc.
- **cocktail party problem:** getting computers to be as good as humans at focusing on a single speaker in a noisy room
- **embedding:** a mapping of high-dimensional features (like points in a spectrogram) to vectors of real numbers in a low-dimensional space
- ***k*-means:** a partitioning of data points into k clusters such that each point belongs to the cluster with the nearest mean
- **objective function:** A measure of error that you want to minimize. When you train a neural network what you are doing is iteratively changing the parameters of the network to minimize the objective function as much as you can.
- **open vs. closed speaker set:** In a closed speaker set, the speakers used in testing were part of the set used for training. In an open speaker set, the speakers in the test set have not been seen before. (So, open is much closer to the real-world problem we want to solve.)

- **oracle:** A model where you give it some or all of the parameters you know to be correct by some other means. This is cheating of course and is not used in the ultimate deployment of a model, but it can be useful when assessing how good the algorithm is under best-case conditions.
- **permutation problem:** This is a challenge specific to source separation. I sort of understand it, but sort of don't.

  Here's one attempt to describe it: There are multiple valid output masks that differ only by a permutation of the order of the sources, so a global decision is needed to choose a permutation.

  Alternative description: "Permutation problem in speech separation arises due to the fact that the order of targets in the mixture is irrelevant. For example, in separating speakers A and B, both (A,B) and (B,A) solutions are acceptable. However, training the neural network with more than one target label per sample will produce conflicting gradients and thus lead to convergence problem, because the order of targets cannot be determined beforehand."
- **spectrogram:** a 2D plot of an audio recording with time on the horizontal axis and frequency on the vertical axis. Used to visualize how the frequency content changes over time. Each point in the spectrogram is a complex number (amplitude and phase). Images of spectrograms usually just show the amplitude using a color scale. Computed via a *short-time Fourier transform (STFT)*.



- **short-time Fourier transform:** convert a time series (like audio) from amplitude-versus-time (time domain) to frequencies (frequency domain) over a series of successive windows. This is done in such a way that the process can be reversed and the original signal recovered.