Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. Hermann Ney

Seminar Titel im SS 2018

Deep Clustering for ANN Supported Source Separation

Patrick von Platen

Matrikelnummer 331 430

Datum des Vortrages

Betreuer: Tobias Menne

Contents

List of Tables

List of Figures

1 Introduction

When perceiving sound created by multiple acoustic sources, the human auditory system is extremely good at focussing on parts of the sound coming from only one acoustic source.

Imagine yourself at a cocktail party. The sound that you perceive is made up of all kinds of acoustic sources: The voice of the person you are listening to, the sound of music, the background chatter of other people talking, the sound of clinking glasses, etc. Still you can clearly understand the speaker in your group, even though its acoustic signal overlaps at every moment with the acoustic signal of other people speaking and background noise.

The difficulty of understanding speech in a multiple speaker environment is often called "The cocktail party problem" [4], first mentioned by Colin Cherry in 1953. It was also Colin Cherry, who first brought up the idea of automatic speech separation in his famous paper on "experiments of the recognition of speech" [6]. The first method to yield some promising results was introduced by A.S. Bregman in 1990 using *Auditory scene analysis* [3]. Since then, multiple methods have been explored. One of them is *Spectral Clustering* , a method based on partitioning data points into clusters according to eigenstructure of a similarity matrix.

With the receent rise of deep learning in a variety of applications, the first method to be applied to source separation was presented by Chao Weng and co. in 2015 [17]. Quickly, different deep learning methods based on a technique called *Deep Clustering* emerged, forming the state-of-art in the source separation problem. This report, will focus on the deep clustering method as it was introduced by John R. Hershey and co. in 2015 [8] and will study *TasNet*, a deep clustering system performing audio separation in the time domain yielding state-of-the art results [10].

The applications of automated source separation are wide-ranging. To name a few:

- *Automatic meeting transcription*: In business meetings, multiple speakers are alteranating in a short time or even speaking at the same time. Making it possible to automatically separate speakers from each other and transcribe would be of great use for companies.

- *Virtual assistant*: Virtual assistants, like Alexa, Siri and Google Home are playing a critical part in smart home systems. Filtering out noise and separating the owner's voice from other voices are challenges that can be taken on by automated source separation techniques.

- *Automatic subtitling of music/video*: According to the World Health Organization (WHO) over 5% of the world population suffers from disabling hearing loss [1]. Subtitling of music and video content, which are often made up of sound coming from multiple acoustic sources, is therefore essential for them.

To begin with, the source separation problem is mathematically defined and basic mathematical operations and basic deep learning methods used in deep clustering are explained 2. In the following conventional methods and first attempts of using deep learning techniques for automated source separation are presented 3. The essence of this paper being deep clustering and *TasNet* will be presented in-detail in section 4. Finally, a conclusion is drawn 5.

2 Basics

This section firstly defines the source separation problem and then gives an overview of the basic operations and methods used in systems.

2.1 Definition source separation problem

A generel definition of the source separation problem was given by Jean-Francois Cardoso, defininig the problem as "recovering unobserved signals or sources from several observed mixtures" [5]. He also added two important restrictions that we will apply to our definition as well:

1. the source signals are not observed
2. no information is available about the mixture

The condition of not knowing how many sources the mixture is composed of, became known as *blind* source separation [5]. In this paper, we add another restriction: The signal is observed over a single channel (single microphone). To sum it up, the *source separation problem* is defined in this paper as "recovering the original signals of an unknown amount of sources from a single-channel mixture of the source signals".

Putting the above definiton in a mathematical form:

- the unknown number of acoustic sources: N.
- the original sound of the i-th source at time t: $x_i(t)$.
- the single received sound signal at time t: $X(t) = \sum_{i=1}^{N} x_i(t)$.

The goal of source separation is to recover $x_1(t), ..., x_N(t)$ from $X(t)$. An algorithm performing this task is presented below.

Input: $X(t)$
Output: $(N, \{x_1(t), ..., x_N(t)\})$

2.2 Mathematical operations

This subsection aims to shine light on essentiel mathematical operations used in the systems that will be explained in sections 3, 4.

2.2.1 Short term fourier transform

In order to understand the short term fourier transform, one firstly has to get an understanding of the fourier transform itself. Having a continous signal $s(t)$, the fourier transform is defined as

$$S(f) = \int_{\mathbb{R}} s(t) e^{-2j\pi ft} dt$$

The fourier transform $S(f)$ gives the magnitude of every frequency in the signal. A signal can be decomposed as the sum of signals consisting only of one frequency (so called sinusoids). Thus, one can say that the higher the magnitude of a frequency of the fourier

transform, the higher is the percentage of the signal being composed of the sinusoid of this frequency. An important aspect to notice is that the fourier transform integrates over the whole time domain of the signal in order to get the exact magnitude of every freuency present in the signal. Thus integrating only over a part of the time domain leads to approximated values of the frequencies magnitudes. On the other hand, a shorter time intervall to integrate over leads to a better time localization of the frequency. This famous trade-off is well-known as the uncertainty principle in signal analysis [12].

The short term fourier transform divides the signal into local sections using a window function $w(\tau) = 0 : \tau \notin [-a, a]$ over which the fourier transform is applied.

$$S(f,t) = \int_{\mathbb{R}} s(\tau)w(\tau - t)e^{-2j\pi f\tau}d\tau$$

Appling the short term fourier transform converts a signal $X(t)$ to multiple (F, T) binsthat can be plotted in a *spectrogram*. It therefore is used to describe the change in frequency over time. Using an appropiate window funciton leads to a good trade off between time localization and frequency approximation.

2.2.2 Embedding space

An embedding is a structure-preserving, injective mapping $f : X \to Y$. Therefore, every object of domain X is mapped to a distinct object in domain Y . In this report, we are interested in embeddings that map to normed spaces meaning that it is possible to measure the similarity of two objects x_1 and $x_2 \in X$ by measuring its distance $d(y_1; y_2) \in \mathbb{R} : y_1, y_2 \in Y$. It is essentially used to convert data to a feature representation where certain properties can be preserved by distance, so that the data is easily comparable by computers.

One famous example are word embedding models, such as *word2vec* that maps words to a vector representation. Assuming that words have multiple degrees of similarity, word embeddings are used to easily calculate the similarity of words on multiple axis [11]. Later, we will see that the function mapping objects from one domain to an embedding space can be trained on artificial neural networks and are very powerful tools for the source separation problem 4.1.

2.3 Deep neural network basics

This subsection aims to clarify basic deep learning methods and architectures used in the sections 3, 4.

2.3.1 Recurrent neural network

Recurrent neural networks are neural networks that are "specialized in processing a sequence of values $x_1, ..., x_t$" [7]. For that reason, they are very well applicable when dealing with acoustic signals having high autocorrelation (high correlation between $s(t)$ and $s(t-k, s(t+k)$ for $k > 0$). Recurrent neural networks are extremely effective when dealing with sequential data. First, they process data of different time instances in one computation step, thus taking full advantage of autocorrelation. Second, they can handle data sequences of variable length in contrast to feed forward neural networks where the input length is determined by the size of the input layer. This is achieved by loops between recurrent units inside the neural network [7]. Unfolding a recurrent unit z of a recurrent

neural network at different time instances leads to so-called hidden copies of the network being connected to each other: $z^0, z^1, ..., z^{t-1}, z^t$ As a result, such a recurrent unit z^t at time t takes both the output of the node leading to it (x^t) and the output of the same recurrent unit of one time instance earlier (z^{t-1}) as input resulting in:

$$z^t = f(Vx^t + Wz^{t-1} + b)$$

with $f()$ being the activation function, V the weights between recurrent unit z and input x, W the weights between two hidden recurrent units z^{t-1} and z^t and b the constant bias term. This basic setup is known to have the vanishing/exploding gradient problem due to an exponential decrease/increase in value of long-term components ($\frac{\partial z^t}{\partial z^{t-k}} \rightarrow$ *const.* $\times$ $W^k \rightarrow 0$ or ∞)[13]. To overcome this problem, Hochreiter and co. introduced a "better" recurrent unit, called long shot term memory.

2.3.2 Long short term memory

To overcome the exploding/vanishing gradients problem, the long short term memory unit introduces an internal state which can be seen as the unit's "memory" [9]. At every time instance t, the unit updates its memory c^t using the memory c^{t-} and the unit's output h^{t-1}, as well as the new inputx^t. c^t in combination with h^{t-1} and x^t is then used to produce the new outputht. The structure of the long short term memory unit is shown in Figure 1. It is important to notice that the long short term memory uses x^t and h^{t-1} to compute four different gates: f,i,g,o which are used to control the flow of information in the unit. Also, we can see that when doing backpropagation, the gradient easily "flows" to earlier hidden units without the weight matrix being applied to it (shown by the red arrow) in 1. This "highway" for the gradient strongly reduces the vanishing/exploding gradient effect and allows the unit to take into account long term dependencies of the input data.

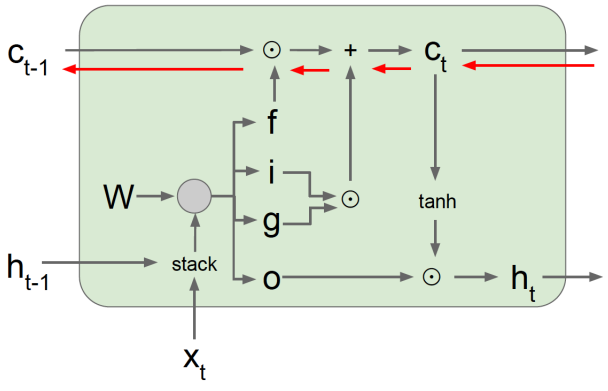

Figure 1: Long short term memory

To furter improve this ability, bi-directional long short term memory units were proposed to exploit correlations between data in both time directions and are used in many state-of-the-art systems nowadays. Going more into detail is out of scope for this report, but more in-detail informationcan be found in [9].

2.3.3 Autoencoder

An autoencoder is a neural network that is "trained to attempt to copy its input to its output" []. It always consists of two parts, the encoder and the decoder. In its simplest form, the encoder maps the input x to one hidden layer $f(x) = z$ and the decoder tries to reconstruct the original data $g(z) = x'$ from this layer. Even though autoencoders are trained with loss functions trying to minimize the difference between x and x', perfect recontruction of the data input is not the goal. Instead, autoencoder should be trained in a way preventing them to just copy the input to the output, but to extract useful information about data distribution instead [7]. Thereby, the hidden layer should represent the most useful information. Two approaches are considered in this report. First, a undercomplete autoencoder is defined by the hidden layer having much lower dimension than the data, thus forcing the network to "store all data in a compressed form". A backdraw of this approach is that the hidden layer might be two small to learn enough useful information. Second, a sparse autoencoder forces the hidden layer to be sparse, by introducing a sparsity penalty on the hidden layer $P(h)$, which will then be added to the loss function $\mathbb{L}(x, x') + P(h)$. Especially in the field of machine translation, autoencoders in combination with recurrent neural networks produce state-of-the-art results [16]. For more detailed information, please refer to [7].

3 Conventional methods

The first theoretical framework for automated source separation was introduced by A.S. Bregman in 1990 [3]. Systems using this approach are called *computational auditory scene analysis* and produced first results, that are acceptable [14]. Section 3.1 will handle cover the system.

Spectral Clustering, a method well-know for data clustering, was then applied to the source separation problem and showed good results (see 3.2). It was the first method used for source separation to train some of its parameters on actual data [2].

3.1 Computational auditory scene analysis

Computational auditory scene analysis for source separation is an automation of the auditory scene analysis which is inspired by the principles of processing sound in human auditory system [14]. In computational auditory scene anlysis the algorithm consists of two parts: *segmentation* and *grouping*. A more ex A detailed description of the algorithm can be seen in the following and it based on the paper of *Wang* and *Brown* [14].

Input: (N, $X(t)$)
Algorithm:

1. Segmentation:
   - X(t) is transformed into time-frequency space using short term fourier transform
   - Segmenation rules are used to define different time-frequency (T,F) regions.
2. Grouping:
   - (F, T) bins are combined to streams corresponding to sound sources
   - Auditory masks are created using different streams

Output: $(\{x_1(t), ..., x_N(t)\})$

The segmentation rules used in the segmentation part are hand-crafted similarity features upon which the different (T,F) regions are created. For every speaker, an audiotry masks is created as a last step of the grouping part. These auditory masks are boolean matrices that can be applied to the spectogram to retrieve the part of the spectogram corresponding to a speaker. Important to notice is that the amount of different speakers N has to be an input of the algorithm and cannot be computed by the system itself. Since computational auditory scene analysis is a rule based system it is very difficult to generelize to broader conditions and needs very precise tuning in order to work.

Computational auditor scene analysis was the first version to produce some meaningful results, but is not powerful enough for the general source seperation problem as we defined it in section 2.1.

3.2 Spectral Clustering

Spectral Clustering is a well-known method in multivariante statistic for the clustering of data. It makes use of the eigenvalues (also called spectrum, thus the name *Spectral clustering*) of a similarity matrix W of the data to reduce the dimensionality of the data. Since spectral clustering will be very important in latter chapters it will be described here in detail. The similarity matrix W can be defined in multiple ways depending on the task, but it should have the following properties.

1. W should be a hermitian matrix: $W^T = W$.
2. $(W)_{i,j} \geq 0, \forall i, j$
3. W should be non-negative definite.

Considering a dataset of P entries $D \in \mathbb{R}^P$, describes the similarity between each of these datapoints, thus $W \in \mathbb{R}^{P \times P}$. Since W can be considered as a distance measure, whereas $W(d_1, d_2) = W(d_2, d_1$ and with only positive distances, the first two of the above defined properties seem reasonable. The third property ensures that all eigenvalues of W are ≥ 0 being a characteristic of non-negative definite matrices.

Having defined a similarity matrix, an example of an algorithm for spectral clustering used in [2] is presented in figure 2.

Input: Similarity matrix $W \in \mathbb{R}^{P \times P}$.
Algorithm:
1. Compute first R eigenvectors U of $D^{-1/2} W D^{-1/2}$ where $D = \text{diag}(W\mathbf{1})$.
2. Let $U = (u_1, \ldots, u_P)^\top \in \mathbb{R}^{P \times R}$ and $d_p = D_{pp}$.
3. Initialize partition A.
4. Weighted K-means: While partition A is not stationary,
   a. For all r, $\mu_r = \sum_{p \in A_r} d_p^{1/2} u_p / \sum_{p \in A_r} d_p$
   b. For all p, assign p to A_r where $r = \arg\min_{r'} \|u_p d_p^{-1/2} - \mu_{r'}\|$

Output: partition A, distortion measure $\sum_r \sum_{p \in A_r} d_p \|u_p d_p^{-1/2} - \mu_r\|^2$

Figure 1: Spectral clustering algorithm that minimizes $J_1(W, E)$ with respect to E with weighted K-mean. See Section 2.6 for the initialization of the partition A.

Figure 2: Spectral clustering algorithm

First, all eigenvalues and their corresponding eigenvectors of the similarity matrix are calculated. The eigenvectors $v_i \in \mathbb{R}^P$ can be sorted according according to their eigenvalues in descending order, such that $v_0, v_1, ..., v_r, ..., v_p$ with $\lambda(v_0) = max_{i \in \{0,...,P\}} \lambda_i$. Now we can choose the first R eigenvectors to have R different clusters. Our set of eigenvectors is defined as $V = (v_0, v_1, ..., v_r) \in \mathbb{R}^{P \times R}$. We can now define a set $U = (u_0, u_1, ..., u_p) = V^T \in\in \mathbb{R}^{R \times P}$, whereas u_i corresponds to exactly one data point and is the vector containing the i-th elements of all eigenvectors $v_1, ..., v_r$. As a last step, we apply a weighted version of the *k-means algorithm* [15] by first randomely selecting R vectors $m_1, ..., m_r$ as mean vectors and then successively assigning the vectors $u_1, ..., u_p$ and updating $m_1, ..., m_r$. After conversion, all vectors assigned to m_j present the j-th cluster.

Bach and Jordan applied spectral clustering for source separation of two-speaker mixtures. The method described in the following is based on their paper "Learning spectral clustering, with application to speech separation" [2].

First, the definition of a similarity matrix W corresponding to the mixture signal $X(t)$ will be derived. As a first step the mixture signal can be convert to (F, T) bins using the short term fourier transform as explained in 2.2.1. The (F, T) are then used to create multiple features, which are based on: *Energy, continuity, common fate cues, pitch estimation, timbre,....* In this way, we create multiple features $s_i(f, t)$ for every data point (f, t) whereas the calculation of $s_i(f, t)$ can make use of neighoring data points.

The features are essentially embeddings making the data points more comparable, so that we can define k similarity matrix corresponding to k features $W_k(i, j) = f_k(s_i, s_j)$ choosing appropriate functions f_k. The final similarity matrix consists of a weighed combination of each similarity matrix $W = W_1^{\alpha_1} \odot W_2^{\alpha_2} \odot ... \odot W_k^{\alpha}$ with $\odot$ being the hadamard product. The similarity matrix can then be inserted into our previously explained spectral clustering algorithm 2 and the two clusters can be defined resulting in a neat spectogram for every speaker. The conversion of the data points to a combined similarity matrix works like front-end processor.

The only thing left to explain is how to define the weigth vector $\alpha \in \mathbb{R}^K$. Having N labeled data sets with the (F, T) bins correspondng to one of two speakers so that every data set has a reference partition E'_n, a partition $E_n(\alpha) = \text{K-Means}(W_n(\alpha))$ can be derived. The *k-means* algorithm is applied to the similarity matrix depending on α and the previously defined features as well as the *k-means* algorithm. Finally, a cost function

$\mathbb{L} = \sum_i^N H(E'_n, E_n(\alpha))$ is used to find the α minimizing the $\mathbb{L}$.

Even though spectral clustering can produce very good results on two-speaker separation [2] it has its limitations. Already for seconds of speech sampled at 5.5 kHz leads to 22000 data points and thus, a similarity matrix having 22000×22000 entries not even applying the short term fourier transform. It is obvious that eigenvalue and eigenvector calculations become very expensive. Additionally, the algorithms relies on "handcrafted features" and uses only shallow learning for the similarity matrices by essentially applying linear regression to optimize the scaling vector α. More speakers means more overlapping speakers per (F, T) bin so that features taking into account multiple data points for calculation contain multiple speakers making it pointless to cluster them. A speaker segmentation would be needed before-hand to create useful features, which would miss the point of creating features to perform the segmentation. Deep clustering as explained in 4.1 tries to solve this problem.

4 Deep clustering methods

I want to present three methods here that more or less can be called deep clustering methods and differentiate them between methods using:

1. the (F,T) bins as input -¿ Frequency Domain Audio Separation Based Methods
2. the raw audio signal as input → Time-Domain Audio Separation Based Methods

Explain challanges:

- Permutation problem
- Output dimension mismatch problem

and state for each method how these challenges can be overcome.

4.1 Deep clustering method - Frequency domain audio separation based methods

Method that assigns contrastive embedding vectors to each time-frequency region. Present in detail as described in [8]. Can be used for unknown arbitrary amount of sources.

4.1.1 Training

Talk about what to pay attention to when training.

4.1.2 Results

talk about what ever

4.2 TasNet - Time domain audio separation based methods

Introduce TasNet which does not create masks for each source, but instead use encoder-decoder framework to model directly the signal in the time-domain. Use [10].

4.2.1 Training

Talk about what to pay attention to when training.

4.2.2 Results

Talk about results of TasNet

5 Conclusion

Write conclusion using results.

References

[1] Millionslivewithhearingloss. `http://www.who.int/pbd/deafness/news/Millionslivewithhearingloss.pdf`. Accessed: 2018-05-05.

[2] Francis R. Bach and Michael I. Jordan. Learning spectral clustering, with application to speech separation. *J. Mach. Learn. Res.*, 7:1963–2001, December 2006.

[3] A.S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. A Bradford book. Bradford Books, 1994.

[4] Adelbert Bronkhorst. The cocktail party phenomenon: A review of research on speech intelligibility in multiple-talker conditions. 86:117–128, 01 2000.

[5] Jean-Francois Cardoso. Blind signal separation: Statistical principles, 2003.

[6] E. Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35, March 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[10] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. *CoRR*, abs/1711.00541, 2017.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[12] Sangnam Nam. An uncertainty principle for discrete signals. *CoRR*, abs/1307.6321, 2013.

[13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, 2013.

[14] J. Rouat. Computational auditory scene analysis: Principles, algorithms, and applications (wang, d. and brown, g.j., eds.; 2006) [book review]. *IEEE Transactions on Neural Networks*, 19(1):199–199, Jan 2008.

[15] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[16] Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. Variational recurrent neural machine translation. *CoRR*, abs/1801.05119, 2018.

[17] C. Weng, D. Yu, M. L. Seltzer, and J. Droppo. Deep neural networks for single-channel multi-talker speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(10):1670–1679, Oct 2015.