Chapter 1

Speech Recognition in a Medical Environment

1.1 Background

Automatic speech recognition (ASR) has been a topic that interested many from an early age. Many consider it to have started in the 1950s with Bell Labs' *Audrey* [Moskvitch, 2017], which was able to do a single-speaker digit recognition. Seventy years later, the ASR technologies have grown so that they are present within our personal homes, with Amazon Alexa and Google Home devices [Fowler, 2018], as they have become more affordable. As the speech technologies advance, they approach the concept of ubiquitous computing (often also known as ambient intelligence), which is highly desirable for many industries. Although most of the ASR systems available for commerce are trained in normal language and not medical language, one industry that could greatly benefit from ASR technologies is the health care system, given that the ASR system is able to understand medical language.

Health care is a system which has a very high demand and the services are required to be as detailed as possible due to various reasons, one of which is the concept that it is paramount for a hospital to have a clear and rich track of a patient's medical history. Currently, in order to maintain a patient's medical history, a physician sees multiple people during his/her working hours, and only after their shift is over, does (s)he sit down to write the medical notes. The lag in between seeing a patient and taking notes may sometimes go up to 8-10 hours, and then those notes are often inaccurate. Having an ASR system in a medical environment could greatly help in keeping track of a patient's medical history, where a physician could easily dictate the notes. In a hospital environment, however, such as in an intensive care unit (ICU), a physician who is trying to dictate notes may find him or herself in trouble, as in the ICU, there are multiple background sounds from machines, multiple people speaking, and a great amount of white noise.

Current companies have been creating ASR systems that can understand medical language and allows physicians to dictate their medical notes. One major company that has been "dominating" a lot of the market is Nuance, with their Dragon Medical system. Unfortunately, their dictation system is not yet capable of inferring punctuation marks and markup language onto the text, thus, in order to dictate a segment such as: "37-year-old female presents complainint of urinary frequency, urgency and dysuria along with hematuria and low-grade fever." needs to be dictated as:

"37-year-old female presents complainint of urinary frequency **comma** urgency and dysuria along with hematuria and low-grade fever **period**"

Although the system has very high accuracy results, the system works best when one is in a quiet environment, which is not always a realistic scenario. The goal of this project is to create a transcription engine that can recognize medical language in a noisy environment.

1.2 Preliminary Data

1.2.1 Comparison of Different Commercial Engines

Compare commercial (+ CMU) Text-¿Voice-¿ Text MIMIC II + 20kLeagues

1.2.2 Grade Reading Level of Various Texts

MIMIC II + 20kLeagues

1.3 Automatic Speech Recognition Schematic for a Medical Setting

In order to have a speech recognition system to be able to understand medical language, it is important to obtain a big picture of the project.

[INSERT FIGURE]

As one can see from the diagram above, the sound signals would come to a microphone or an array of microphones, which would then be passed through a system that performs blind source separation on the sound signal. It is very possible that, in a hospital environment, there may be multiple sources, such as the heart rate monitor, the ventilator, the healthcare providers talking, and others. A blind source separation algorithm would ensure to separate the audio channel to multiple sources. Once the sources have been separated, there would be a classification methodology to identify the physician from the patient, and ...

1.3.1 AI-Complete and ASR-Complete

Although there have been many advancements in the speech technologies, especially through black boxes such as neural networks, and there currently exists very accurate engines to do voice recognition, it still remains to be an unsolved problem. The speech recognition problem is ASR-complete, which falls under the umbrella of AI-complete, which, by analogy of NP-completeness, means that the problem is hypothesized to deal with multiple parts of the AI world, and it cannot be solved by a single algorithm. Current technologies are not able to solve AI-complete problems, and they often require human computation. Since speech recognition is an AI-complete problem, there is still a lot of room for improvement in the fundamental problems that have not yet been solved, and although deep learning has recently shown promising results, deep learning neural networks (DNN) do not necessarily solve fundamental problems, as no one has a truthful understanding of how they work and behave. Therefore, there are a lot of possibilities for the formulation design of the framework (i.e. pipeline).

1.4 Speech Enhancement

Speech enhancement is one of fundamental problems of speech that has not yet been fully solved yet, as noise is subjective, and there isn't necessarily one specific way of performing a fully accurate noise reduction for any type of noise. As previously mentioned before, a ventilator and heart rate monitor are examples of background noise that do not necessarily have a Gaussian distribution (i.e. it is not a white noise), yet, for ASR, they are both irrelevant. There are a number of methodologies that have been created to attempt to solve the problem of speech enhancement, and they may potentially be useful to be implemented onto the pipeline.

1.4.1 Recurrent Neural Networks for Noise Suppression

The work on recurrent neural networks (RNN) was originally worked sequentially by [Jordan, 1986] and [Barak, 1988], and it finally came to be coined by [Cleeremans et al., 1989]. They have shown to work very efficiently with time-series data as their cyclic nature allows them to adapt to incoming inputs in a sequential form, working often times better than convolutional neural networks (CNN), when dealing with time-series data. Other variants of RNNs have been created, which allow them to store "memory" for future cases, with Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997], which were then further expanded with gates for Gated Recurrent Units (GRU) [Gers et al., 2000] specifically to allow such units to forget information, and a visual representation of LSTMs and GRUs are shown on Figure 1.1

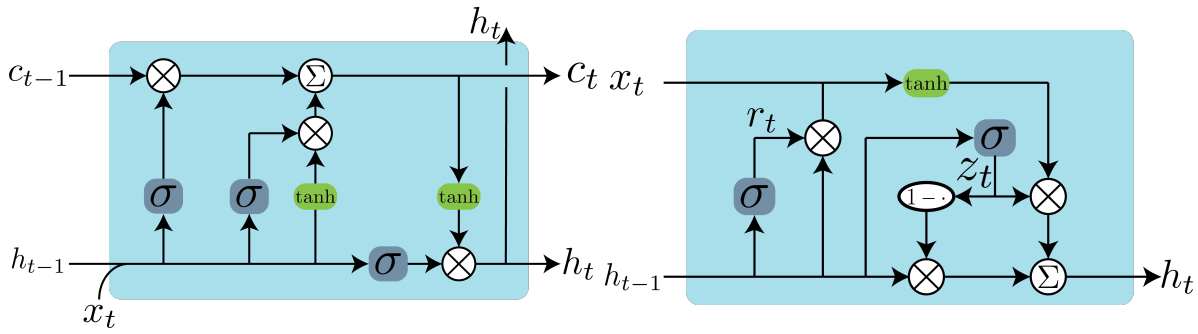

Figure 1.1: A visual representation of the LSTM cell (Left) and the GRU cell (Right). The LSTM contains three representations of its states: the cell state/memory c_t, the hidden states h_t, and the input state x_t, all of which occurs at time t. On the GRU, there are the input state x_t, the hidden state h_t, the update gate z_t and the reset gate r_t. The σ block is used to represent a sigmoid function, and the tanh block is used to represent a hyperbolic tangent for both images. Please see **??** for the mathematical expressions of the cells above

Because of the nature of the network being able to receive continuous sequential inputs, it becomes a very attractive model to potentially behave as an adaptive filter. Hence, members from the Xiph.Org Foundation, a non-profit organization, along with Mozilla have created a RNN architecture that is based on GRUs called RRNoise [Jean-Marc et al., 2017], which, after trained, is able to reduce a lot of the background noise of audio files, thus enhancing the signal. Below on Figure 1.2 is the topology of the RRNoise architecture, where it outputs voice activity detection (VAD) as well as the gains from the input features.

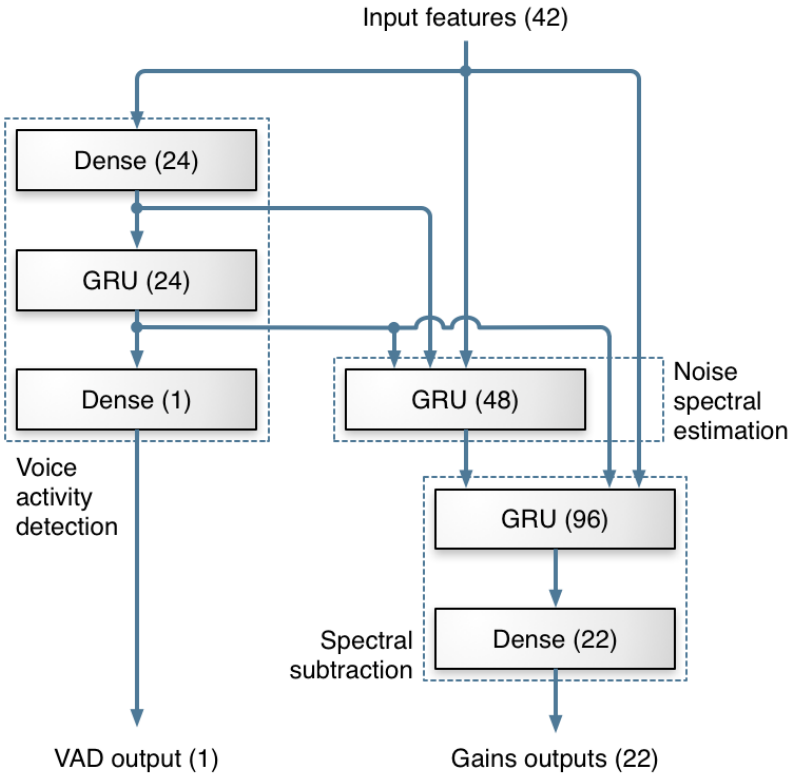

Figure 1.2: The topology of the RRNoise network used to perform noise reduction

1.4.2 Non-Negative Matrix Factorization

Another very neat way of performing speech enhancement is via non-negative matrix factorization. Please skip to the Blind Source Separation's subsection on Non-Negative Matrix Factorization [Ng et al., 2001]

1.5 Blind Source Separation

Given that a recording seldom contains one and only one speaker speaking at a time, an ASR may find itself troubled in detecting who is the speaker and who should be taken as the background. This is namely known as the Cocktail Party Effect. Its name originates from the idea that, in a cocktail party, there are multiple people speaking to one another. While a human is able to have a conversation with another, focusing on his/her friend while having all of the other speakers (i.e. sources) as background, an ASR system is not able to do so. Hence it is important to be able to separate a mixed signal to then be able to focus on a single individual and this is called blind source separation (BSS). There are two methodologies that are proposed to perform BSS: independent component analysis, and non-negative matrix factorization.

1.5.1 Independent Component Analysis

1.5.2 Non-Negative Matrix Factorization (NMF) and Sparse NMF (SNMF)

Background

One of the major drawbacks from any methodology in ICA is that it requires N or more observations (i.e. microphone recordings) for N sources (e.g. speakers, noise), which, depending on the circumstance, may be limiting. The method of non-negative matrix factorization (NMF), however, is able to perform BSS with very high accuracies, while using a single-channel recording. NMF originated in the field of chemometrics, where chemists were trying to identify contents of solutions via positive matrix factorization [Paatero and Tapper, 1994] [Anttila et al., 1995]. It was later further developed in [Lee and Seung, 1999] and algorithms were then better formulated in [Lee and Seung, 2000].

The concept behind NMF is that a mixed signal should be able to be decomposed into its unmixed signals, under the assumption that all of the superpositioned signals are non-negatively added to one another. In other words, they are positively added, but their value may also be zero. Conceptually, this makes absolute sense. If one see the work in [Lee and Seung, 1999], one will see that many of the components added are interpretations of existing physical features, such as eyebrows and mustache. It does not make sense to add a negative eyebrow to reconstruct one's face. Hence, in retrospect, it makes complete sense why this methodology was being used in chemometrics, as it does not make sense to add a negative sulfate ion to a solution.

One may ask how is this factorization different from other types of factorizations, such as principal component analysis (PCA). First, every factorization has different purposes; for instance, a Cholesky decomposition was specifically made to obtain lower computation complexity in order to invert matrices, where as PCA was made to obtain information about its principal orthogonal components. Second, when looking at the comparison between a signal reconstruction or separation from an NMF and from PCA, PCA poses a lot of problems. PCA only allows for the feature space to represent vectors that are orthogonal, which is not always the case. In the sense of speech, it is almost impossible for one speech signal to be orthogonal to another one. PCA also assumes that the probability distribution of the data follows a multivariate Gaussian, which again is not always the case. A NNF is very promising as it performs a matrix decomposition, which is very desirable for computational reconstruction in the last step of the BSS, and it is able to attain reconstructions just as detailed as some of the other reconstruction techniques, plus one is allowed to determine how many components to use.

Here, the notation follows the notations shown in [Schmidt and Olsson, 2006].

Consider the magnitude of a spectrogram of a mixed speech signal $\boldsymbol{Y} \in \mathbb{R}^{M \times N}$, which is a summation of R source signals (i.e. $\boldsymbol{Y} = \sum_{i=1}^{R} \boldsymbol{Y}_i$). The spectrogram can be sparsely represented in an overcomplete basis as

$$\boldsymbol{Y} = \boldsymbol{D}\boldsymbol{H}$$

where $\boldsymbol{D} \in \mathbb{R}^{M \times R}$ represents a compendium of dictionaries with R columns, which can be chosen by the user, and $\boldsymbol{H} \in \mathbb{R}^{R \times N}$ represents a code matrix, which is sparse and it contains code matrices associated with the

dictionaries in $\boldsymbol{D}$. The matrix $\boldsymbol{D}$ and $\boldsymbol{H}$ can be represented as

$$\boldsymbol{D} = \begin{bmatrix} | & | & & | \\ | & | & & | \\ | & | & & | \\ \boldsymbol{D}_1 & \boldsymbol{D}_2 & \cdots & \boldsymbol{D}_R \\ | & | & & | \\ | & | & & | \\ | & | & & | \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} - - - & - - - & - - - & \boldsymbol{H}_1 & - - - & - - - & - - - \\ - - - & - - - & - - - & \boldsymbol{H}_1 & - - - & - - - & - - - \\ & & & \vdots & & & \\ - - - & - - - & - - - & \boldsymbol{H}_R & - - - & - - - & - - - \end{bmatrix}$$

A simple version of how a NMF would work is shown in Figure 1.3, where the figure below was inspired on the works from [Schmidt and Olsson, 2006].

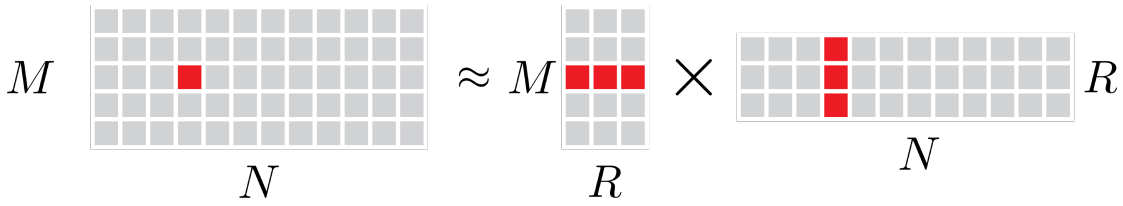

Figure 1.3: A very simple schematic of a non-negative matrix factorization, where the matrix on the left represents the mixed signal, the matrix on the center represents the dictionary matrix, and the matrix on the right represents the coding matrix

There are two different ways to obtain the desired decomposition, and that is based on what optimization approach one wishes to take. One approach is to minimize the squared Euclidean-based Frobenius norm, $\|\boldsymbol{Y} - \boldsymbol{DH}\|_F^2$, and another approach is to minimize the divergence, $\mathcal{D}(\boldsymbol{Y}\|\boldsymbol{DH})$, both cases with respect to $\boldsymbol{W}$ and $\boldsymbol{H}$, subject to the contraints that $\boldsymbol{D}, \boldsymbol{H} \geq 0$ [Lee and Seung, 2000]. Both of their definitions are shown below, where the subscript i represents the i^{th} row, and the j represents the j^{th} column. Notice that the divergence measurement below is not the Kullback-Leibler (KL) divergence, but it does converge to the KL divergence if $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$s

$$\begin{cases} \text{Norm:} & \|A - B\|_F^2 = \sum_{i,j} (A_{ij} - B_{ij})^2 \\ \text{Divergence:} & \mathcal{D}(A\|B) = \sum_{i,j} \left(A_{ij} \log \left(\frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right) \right) \end{cases}$$

Because this is an iterative algorithm, if one follows the norm minimization, the update rules are:

NMF Algorithm for Norm Minimization

1.Initialize $\boldsymbol{D}, \boldsymbol{H} \geq 0$

2.$\boldsymbol{H}_{ij} \leftarrow \boldsymbol{H}_{ij} \frac{(\boldsymbol{D}^T \boldsymbol{Y})_{ij}}{(\boldsymbol{D}^T \boldsymbol{DH})_{ij}}$

3.$\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \frac{(\boldsymbol{Y}\boldsymbol{H}^T)_{ij}}{(\boldsymbol{DHH}^T)_{ij}}$

4.Repeat 2,3 until stability

If one follows the divergence minimization, the update rules are

NMF Algorithm for Divergence Minimization

1.Initialize $\boldsymbol{D}, \boldsymbol{H} \geq 0$

2.$\boldsymbol{H}_{ij} \leftarrow \boldsymbol{H}_{ij} \frac{\sum_k \boldsymbol{D}_{ki} \boldsymbol{Y}_{kj} / (\boldsymbol{DH})_{kj}}{\sum_l \boldsymbol{D}_{li}}$

3.$\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \frac{\sum_k \boldsymbol{H}_{jk} \boldsymbol{Y}_{ik} / (\boldsymbol{DH})_{ik}}{\sum_l \boldsymbol{H}_{jl}}$

4.Repeat 2,3 until stability

If one performs an NMF, a visual representation is shown on Figure 1.4, where the figure below was inspired on the works from [Schmidt and Olsson, 2006].

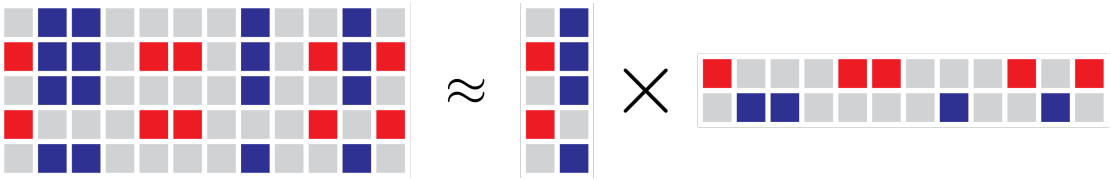

Figure 1.4: A visual example of an NMF result, with two sources being separated.

The algorithms for NMF were further improved by [Eggert and Korner, 2004], where the authors created a Sparse NMF. Their work was inspired by the basic NMF problems that, when one had overcomplete dictionaries, there was no well defined solution such that you would obtain sparse solutions. Thus, they wanted to make sure to obtain sparse solutions, especially on the coding matrix. To tune the sparsity measurement, the authors simply applied a L1 regularization based on the coding matrix coefficients on the cost functional of the norm, and called it a Sparse NMF. In other words, the cost (or energy) functional that originally was $\mathcal{E} = \|Y - \bar{\boldsymbol{D}}\boldsymbol{H}\|_F^2$ became

$$\mathcal{E} = \|\boldsymbol{Y} - \bar{\boldsymbol{D}}\boldsymbol{H}\|_F^2 + \underbrace{\lambda \sum_{i,j} H_{ij}}_{L1 \text{ regularization}} \qquad s.t. \quad \boldsymbol{D}, \boldsymbol{H} \geq 0$$

where $\bar{\boldsymbol{D}}$ still follows the notation of [Schmidt and Olsson, 2006], representing a column-wise normalized dictionary matrix, and λ is the parameter to control the degree of sparsity. To solve this optimization problem, if one was to set $\boldsymbol{R} = \boldsymbol{DH}$, then one would obtain the following algorithm. Note, for this one case, that the dot sign $\cdot$ and and division operator $\frac{a}{b}$ represent point-wise multiplication and division

Sparse NMF Algorithm for Frobenius Norm L1 Regularized Minimization

1.Initialize $\boldsymbol{D}, \boldsymbol{H} \geq 0$

2.$\boldsymbol{H}_{ij} \leftarrow \boldsymbol{H}_{ij} \cdot \frac{\boldsymbol{Y}_i^T \bar{\boldsymbol{D}}_j}{\boldsymbol{R}_i^T \bar{\boldsymbol{D}}_j + \lambda} \in \mathbb{R}^1 \cdot \frac{\mathbb{R}^{1\times M}\mathbb{R}^{M\times 1}}{\mathbb{R}^{1\times M}\mathbb{R}^{M\times 1} + \mathbb{R}^1}$

3.$\boldsymbol{D}_{ij} \leftarrow \boldsymbol{D}_{ij} \cdot \frac{\sum_i \boldsymbol{H}_{ij}[\boldsymbol{Y}_i + (\boldsymbol{R}_i^T \bar{\boldsymbol{D}}_j)\bar{\boldsymbol{D}}_j]}{\sum_i \boldsymbol{H}_i j[\boldsymbol{R}_i + (\boldsymbol{Y}_i \bar{\boldsymbol{D}}_j)\bar{\boldsymbol{D}}_j]} \in \mathbb{R}^{M\times 1} \cdot \frac{\sum_i \mathbb{R}^1 + (\mathbb{R}^{1\times M}\mathbb{R}^{M\times 1})\mathbb{R}^{M\times 1}}{\sum_i \mathbb{R}^1[\mathbb{R}^{M\times 1} + (\mathbb{R}^{1\times M}\mathbb{R}^{M\times 1})\mathbb{R}^{M\times 1}]}$

4.Repeat 2,3 until stability

All of the algorithms shown above are the steps necessary to train the algorithm, where $\boldsymbol{Y}$ contains all of the training data. In order to actually perform a speech reconstruction, one should simply perform the algorithms above, but to keep the dictionary matrix $\boldsymbol{D}$ fixed and update only the code matrix $\boldsymbol{H}$. The speech is then separated by conputing the reconstruction of parts of the sparse decomposition. In other words:

1. Apply NMF or Sparse NMF to learn the dictionaries of individual speakers

2. To separate the mixtures, keep $\boldsymbol{D}$ fixed, and only update $\boldsymbol{H}$

3. Reconstruct the signal by using the desired parts of the decomposition

Aproaches to Learn Dictionaries

As shown in [Schmidt and Olsson, 2006], there are multiple approaches to perform BSS with NMF or SNMF: unsupervised approach and segmenting the training data and solving multiple NMF problems

In the unsupervised approach, one would compute the SNMF over a very large dataset matrix $\boldsymbol{Y}$ of a single speaker to obtain a single dictionary. This, however, may take a lot of computation time. The second approach, used in [Schmidt and Olsson, 2006] involved segmenting the training data according to phoneme labels obtained by a speech recognition software with hidden-Markov models (HMM) for a single speaker, and creating a sparse dictionary for each phoneme, and finally , a final dictionary would be constructed by the concatenation of individual phoneme dictionaries. In order to create the dictionary of each phoneme, each type of phoneme would be concatenated together, to create that phoneme's dictionary, and then, the phonemes dictionaries would be concatenated together to create a dictionary that is representative of the speaker. Figure 1.5 shows an illustration on how to create the dictionary.

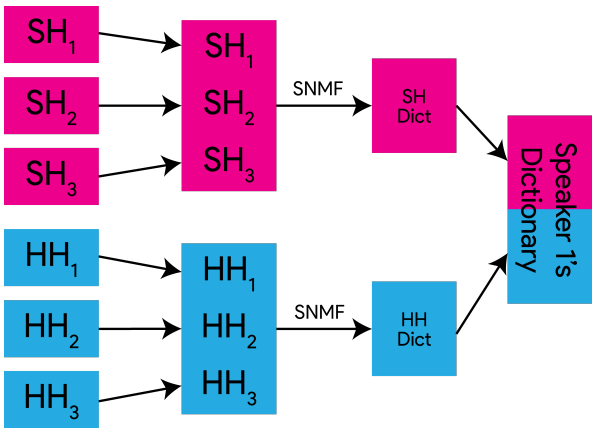

Figure 1.5: An illustration on how to create a speaker's dictionary $\boldsymbol{D}_i$

NMF for Speech Enhancement

The works from [Li and Xiao,] have shown that it is also possible to perform speech enhancement on a speech signal via non-negative matrix factorization. An illustration on Figure 1.6, inspired by the images on [Li and Xiao,], shows the procedure. It is possible to generate the dictionary matrix $\boldsymbol{D}$ and the coding matrix $\boldsymbol{H}$ from a noisy signal, however one would not know which of the subdictionaries $\boldsymbol{D}_i$ or the code submatrices $\boldsymbol{H}_i$ represent the ones associated with the speech and with the noise. Through spectral clustering (for technical details, see **??**), it is possible to group the subparts of the dictionary matrix and the code matrix so that it is differentiated between speech and noise. Given that the reconstruction process is based on linear algebra rules, it is then possible to simply multiply the dictionary matrix related to speech $\boldsymbol{D}_{speech}$ and the code matrix related to speech $\boldsymbol{H}_{speech}$ to obtain the cleaned speech.

Of course this solution is not guaranteed to remove every single noise source due to the possible stochasticity that is propagated to the eigenvectors of the Laplacian, and the fact that the solution is not unique, which is based on the initialization of the centroids of the k-means algorithm, but it could be a good fundamental method to implement.

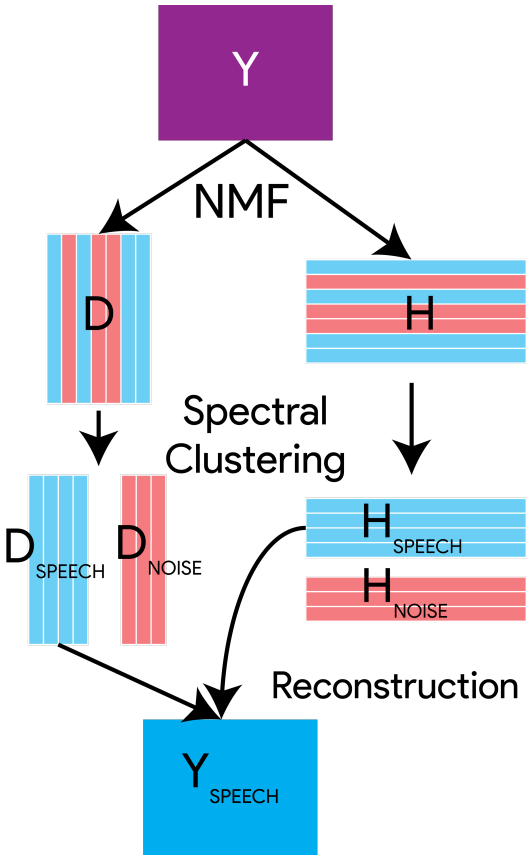

Figure 1.6: An illustration on how to use non-negative matrix factorization to denoise a signal, where $\boldsymbol{Y}$ represents the noisy signal, $\boldsymbol{D}$ represents the dictionary matrix obtained through NMF, and $\boldsymbol{H}$ represents the code matrix, also attained through the NMF.

Knowledge comes from finding the answers. Yes, but understanding what the answers mean is what brings wisdom.

Bibliography

[Anttila et al., 1995] Anttila, P., Paatero, P., Tapper, U., and Järvinen, O. (1995). Source identification of bulk wet deposition in finland by positive matrix factorization. *Atmospheric Environment*, 29(14):1705 – 1718.

[Barak, 1988] Barak, A. P. (1988). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269.

[Cleeremans et al., 1989] Cleeremans, A., Servan-Schreiber, D., and McClelland, J. L. (1989). Finite-state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381.

[Eggert and Korner, 2004] Eggert, J. and Korner, E. (2004). Sparse coding and nmf. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, volume 4, pages 2529–2533 vol.4.

[Fowler, 2018] Fowler, G. (2018). I live with alexa, google assistant and siri. here's which one you should pick. *The Washington Post: The Switch Review*.

[Gers et al., 2000] Gers, F. A., Schmidhuber, J., and Cummins, F. A. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12:2451–2471.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Jean-Marc et al., 2017] Jean-Marc, Xiph.Org, and Mozilla (2017). Rrnoise.

[Jordan, 1986] Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist seuential machine. *Cognitive Science Conference*, pages 531–546.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee and Seung, 2000] Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA. MIT Press.

[Li and Xiao,] Li, P. and Xiao, Y. Matrix factorization for speech enhancement.

[Moskvitch, 2017] Moskvitch, K. (2017). The machines that learned to listen. *BBC Future*.

[Ng et al., 2001] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press.

[Paatero and Tapper, 1994] Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

[Schmidt and Olsson, 2006] Schmidt, M. N. and Olsson, R. K. (2006). Single-channel speech separation using sparse non-negative matrix factorization. In *INTERSPEECH*.