Math 444: Project 5

Levent Batakci

April 15, 2021

In this project, we leveraged non-negative matrix factorization to separate time-dependent signals originating from different sources. This is one of the many ways to approach the Blind Signal Separation (BSS) problem. All of the mentioned implementations can be found on the public github repo.

Non-negative Matrix Factorization Background

Non-negative Matrix Factorization (NMF) is a way to factor a matrix $X \in \mathbb{R}^{n\times p}$ into a product of two matrices $W \in \mathbb{R}^{n\times r}$ and $H \in \mathbb{R}^{r\times p}$. It's important to note that such a factorization is not guaranteed for $r < p$. This might seem problematic, but in most cases we can approximate $X \approx WH$ relatively well.

The importance and utility of this factorization stems from how we can interpret it. Generally, the columns of W or rows of H provide insight on the data. Note that for non-negative data, it makes the most sense to have non-negative feature vectors. Singular value decomposition, which is powerful in its own regard, is practically incapable of this.

It should be noted that the NMF of matrix is not unique. Indeed the product WH remains the same if the columns of W are scaled by α and the rows of W are scaled by $\frac{1}{\alpha}$ for any $\alpha \neq 0$. Of course, to maintain non-negativity, we only consider $\alpha > 0$.

We take an iterative approach to computing the NMF. Specifically, we use multiplicative updating to update W and H until neither changes a significant amount. We do not use nearness to X as a stop condition as this is not guaranteed. Our stop condition is

$$\frac{||W^{(t+1)} - W^{(t}||_F}{||W^{(t)}||_F} + \frac{||H^{(t+1)} - H^{(t}||_F}{||H^{(t)}||_F} < \tau$$

where $\tau > 0$ is a constant. The matrices $W^{(t)}$ and $H^{(t)}$ represent the approximation of W and H at time step t, respectively.

Experimental Setup

In our experimental setup, we placed 4 sound sources with distinct signals inside of the unit disc. None of the sources were placed in the same spot.

We denote $f_j(t)$ as the function representing the sound emitted from source j. Moreover, the sound sources $f_j(t)$ are scaled so that $0 \leq f_j(t) \leq \max f_j(t) = 1$ for all $1 \leq j \leq 4$.

We uniformly distributed 20 microphones along the boundary of the unit disc. These microphones recorded data at each of p-many uniformly spaced times. A visual representation of the setup is shown in **Figure 1** on the next page.

Here, the red squares represent the 4 sound sources. The green squares represent the microphones, and the black circle represents the boundary of the unit disc.

There is one important idealization to note - we are abusing that sound travels incredibly fast. Since no source is significantly far from any microphone, we are treating sound as travelling instantly.

Indeed, consider the case a single source with 2 microphones. If one microphone is essentially right next to the source and the other is significantly far, the second microphone would essentially record a time-shifted signal, which (under the assumption of instantaneous travel) would lead us to conclude that there are two sources. However, this concern is insignificant in the context of this experiment.

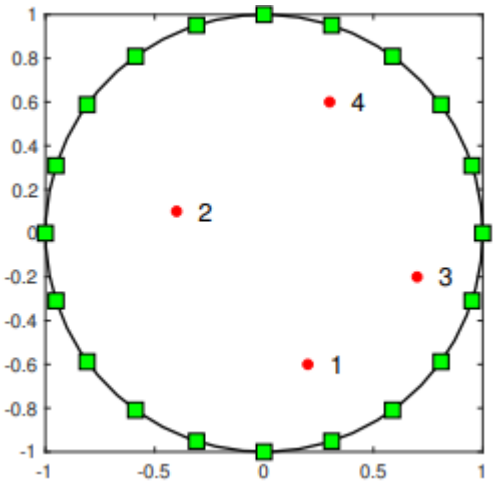

Figure 1: Experimental Setup

Moreover, we are assuming the disc to be filled with material which absorbs the sound, so that it decays even more than usual with distance. The absorption coefficient is $\lambda \geq 0$. We denote $r_{\ell j}$ as the distance between the j-th sound source and the ℓ-th microphone. Also, $\epsilon^{(\ell)}(t)$ is the noise at the ℓ-th microphone at time t. Then, the sound recorded by microphone ℓ is

$$b^{(\ell)}(t) = \epsilon^{(\ell)}(t) + \sum_{1 \leq j \leq 4} \frac{e^{-\lambda r_{\ell j}}}{r_{\ell j}} f_j(t).$$

Our data is encoded in a matrix $X \in \mathbb{R}^{20 \times p}$ where each row represents the discrete evaluations of one of the $b^{(\ell)}(t)$. Note then that in the case of no noise, we can actually compute a perfect NMF of X.

Indeed, letting the j-th row of H be the discrete evaluations of $f_j(t)$ and letting $W_{\lambda j} = \frac{e^{-\lambda r_{\ell j}}}{r_{\ell j}}$. We record the sound signals at $p = 1000$ discrete times. Thus, we can write $X = WH$ where $W \in \mathbb{R}^{20 \times 4}$ and $H \in \mathbb{R}^{4 \times 1000}$. In our analysis, we will look at the rows of H to see if our NMF was able to separate the signals well. It is important to know that in Blind Signal Separation, we know neither the positions of the sources nor the amount of sources.

Since this is an experimental setup, we know the exact signals produced by the sources. **Figure 2** below shows these signals.

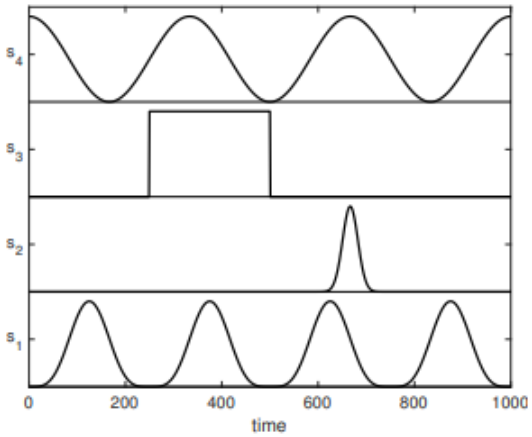

Figure 2: Actual Signals

Non-blind Signal Separation

Before attempting to blindly separate the signals, we decided to compute the NMF with $r = 4$, the correct guess. Essentially, these results should tell us whether the noise ϵ is insignificant enough that we can separate the signals fairly well. We used $\tau = 0.0001$ to determine our stop condition.

Moreover, we leveraged our knowledge that $0 \leq f_j(t) \leq \max f_j(t) = 1$ along with the fact that any NMF is non-unique. That is, we scaled the rows of H to have a maximum of 1. To adjust W, we scaled the corresponding column by the inverse of the respective coefficient.

As discussed in the previous section, in the absence of noise, there exacts an exact factorization of the sound data where the rows of H can represent the signals at the time steps. For this reason, we plotted each of the rows of H. The resulting graphic can be seen in **Figure 3** below.

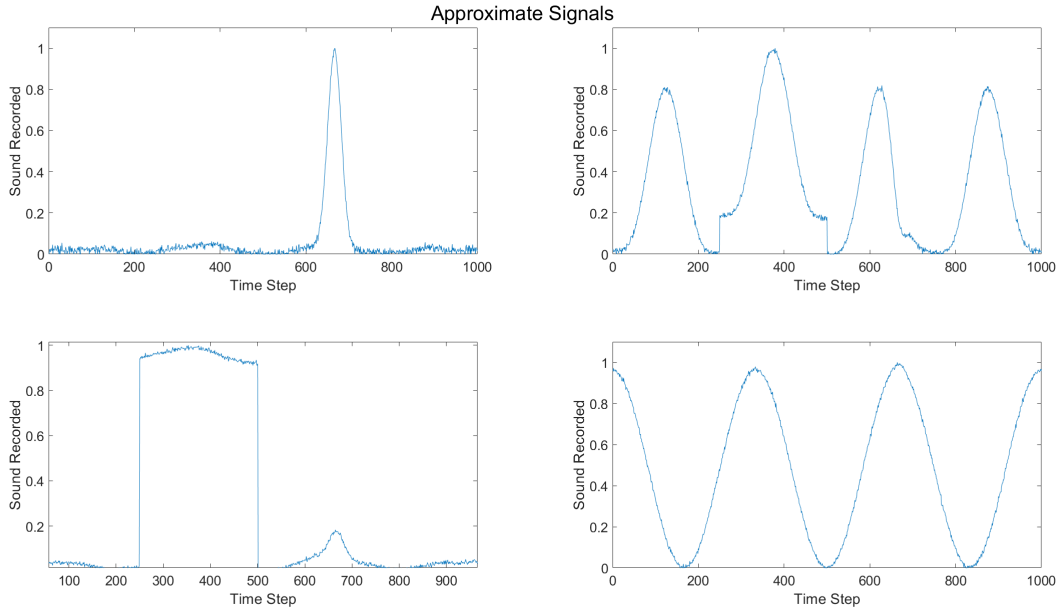

Figure 3: Approximate Signals from NMF with $r = 4$

Immediately, it's apparent that while there are some errors and noise in the approximations, the signals are unambiguously separated. Moreover, comparing the signals to the exact ones (shown in **Figure 2**), the approximations are rather good! Each of the actual signals is clearly present as one of the approximations. Moreover, we computed $||X - WH||_F \approx 2.54$. This leads us to conclude that the noise is relatively manageable and so NMF is very effective at separating these signals.

Blind Signal Separation

In practice, we rarely know the amount of sources or their positions. This leads to us having to guess the correct value of r. To explore the effects of under and over estimating r, we computed NMF with $r < 4$ and $r > 4$.

First, we tested $r = 3$ and $\tau = 0.0001$, which is as close an underestimate to the correct value (4) as we can get. Essentially, we are asking NMF to decompose data generated by 4 signals into 3. As such, we should expect a sizable error in the approximation. We used the Frobenius norm to judge the quality of the approximation. We computed $||X - WH||_F \approx 8.51$ for $r = 3$. Immediately, we notice that this is significantly larger than the error for $r = 4$.

We also chose to plot the resulting rows of H. **Figure 4** below shows these plots. While 3 of the actual signals seem to be present, one is missing altogether.

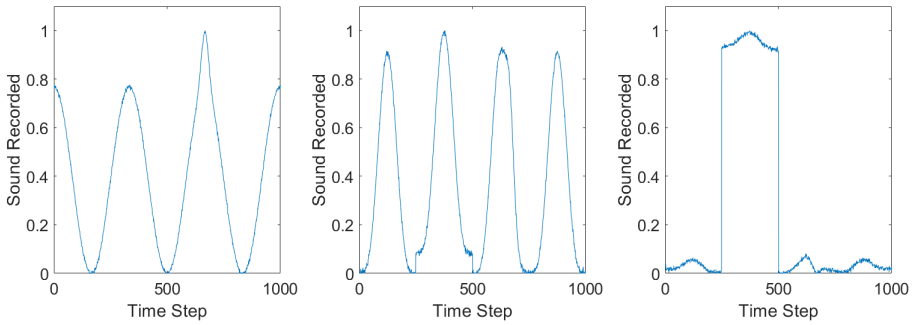

Figure 4: Approximate Signals from NMF with $r = 3$

Next, we tested over-estimating r. Specifically, we used $r = 6$ and a tolerance of $\tau = 0.0001$. Overestimating r, we are asking the NMF to identify more sources than there are. This forces the NMF to split some sources in an arbitrary way. Ultimately, this leads to the approximated signals being much noisier. **Figure 5** below shows this clearly.

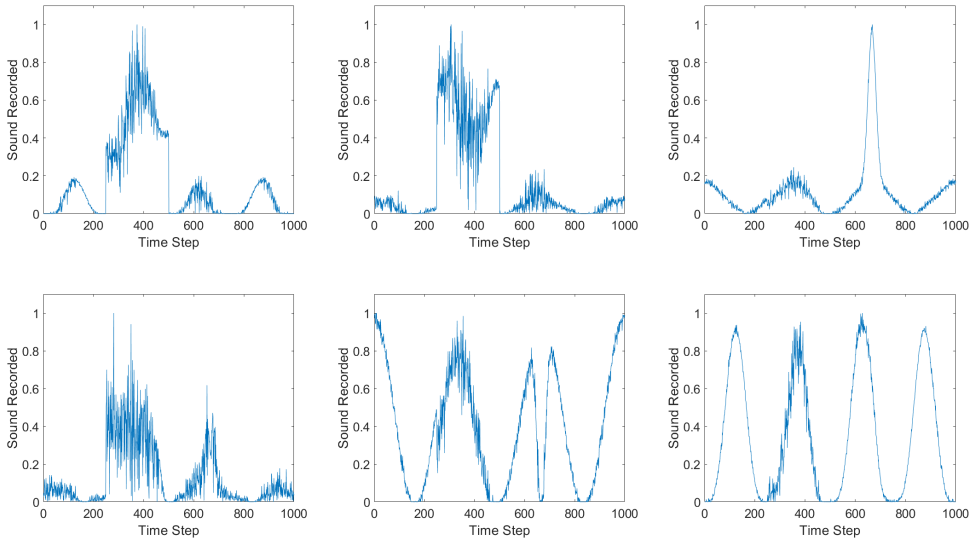

Figure 5: Approximate Signals from NMF with $r = 6$

We can use these results about over and under estimates on r to design a way to automatically guess the optimal r. Indeed, we can start at $r = 1$ and increment it until we get to a point after which the change in the Frobenius norm is sufficiently small and the noisiness of the data increases.

Quantification of noisiness is a complex subject. We propose the following approach. For each row of H, calculate the relative noisiness by computing the sum of residuals of an interpolated approximation of the row. Of course, the interpolation would need to be done with a proper subset of points of the row. In practice, we would use a spline to interpolate. Then, we would compute the average noisiness of each row to be the noisiness of the approximation. For $r > r_{\text{optimal}}$, we should find a higher values of noisiness.

Making such a process automatic is crucial. In practice, one may have to deal with many sources, and decisions may need to be made quickly. A person simply cannot sift through hundreds of values of r in the time frame that modern problems may require. Thankfully, technology propels us past our limits.

Appendix

As always, feel free to access the code at the public github repo.

1 NMF Code

```
function [W, H] = NMF(X, r, tol)
% X approx= WH

[n,p] = size(X);

t = 0;

Wo = rand(n,r);
Ho = rand(r,p);
W = Wo;
H = Ho;

while t==0 || (norm(W - Wo, 'fro')/norm(Wo, 'fro') + norm(H - Ho, 'fro')/norm(Ho, 'fro') > tol)
    %Move forward!
    Wo = W;
    Ho = H;

    %Wo is W^(t), same for Ho
    %W is W^(t+1), same for H

    %Update W
    Xc = Wo * Ho; %approximation of X
    W = ((X*Ho') ./  (Xc * Ho')) .* Wo;

    Xc = W * Ho; %approximation of X
    H = ((W'*X) ./ (W'*Xc)) .* Ho;

    t=t+1;
end

end
```

2 Data Analysis Code

```
%Levent Batakci
%4/6/2021
%Assignment 5 - NMF and blind signal separation

%Load the data
load SoundSourceData.mat

%Plot the actual signals
figure(1)
```

```
sgtitle("Actual Signals");
subplot(2,2,1);
plot(F(1,:));
subplot(2,2,2);
plot(F(2,:));
subplot(2,2,3);
plot(F(3,:));
subplot(2,2,4);
plot(F(4,:));
%These are what we are hoping to find through the NMF


%Use NMF to try to find the signals
%Initially, we will try r=4 (a correct "guess")
r=4;
tol=0.0001;

[W, H] = NMF(X, r, tol);
[W, H] = Rescale(W,H); %Scale the rows of H

figure(2)
sgtitle("Approximate Signals", "FontSize", 30);

subplot(2,2,1);
plot(H(1,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,2,2);
plot(H(2,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,2,3);
plot(H(3,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,2,4);
plot(H(4,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
```

```
set(gca,"FontSize", 20);

%Size of error
err4 = norm(X - W*H, "fro")

%r too small
r=3;
[W, H] = NMF(X, r, tol);
[W, H] = Rescale(W,H); %Scale the rows of H
err3 = norm(X - W*H, "fro")
figure(3)

subplot(1,3,1);
plot(H(1,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(1,3,2);
plot(H(2,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(1,3,3);
plot(H(3,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

%r too big
figure(4)
r=6;
tol=0.0001;
[W, H] = NMF(X, r, tol);
[W, H] = Rescale(W,H); %Scale the rows of H
err5 = norm(X - W*H, "fro")
subplot(2,3,1);
plot(H(1,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,3,2);
plot(H(2,:));
```

```
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,3,3);
plot(H(3,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,3,4);
plot(H(4,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,3,5);
plot(H(5,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);

subplot(2,3,6);
plot(H(6,:));
xlim([0 1000]);
ylim([0 1.1]);
xlabel("Time Step");
ylabel("Sound Recorded");
set(gca,"FontSize", 20);
```