Master Degree in Artificial Intelligence
Statistical and Mathematical Methods for Artificial Intelligence

2021-2022

Non-Negative Matrix Factorization (NMF)

**Blind Signal Separation (BSS) Problem solved with NMF.**

In this exercise, you are given a file named SoundSourceData.mat, contained into the folder data. You are asked to apply NMF to that problem.

We want to use NMF to solve the so-called Blind Signal Separation (BSS) Problem. In particular, we will consider the two dimensional case in which we have $m$ sound sources inside a unit disc. The position of the sources are given by the two dimensional vectors

$$y^{(1)} = (y_1^{(1)}, y_2^{(1)}), y^{(2)} = (y_1^{(2)}, y_2^{(2)}), ..., y^{(m)} = (y_1^{(m)}, y_2^{(m)})$$

The emitted sound is collected by a series of n microphones, located at certain coordinates

$$x^{(1)} = (x_1^{(1)}, x_2^{(1)}), x^{(2)} = (x_1^{(2)}, x_2^{(2)}), ...x^{(n)} = (x_1^{(n)}, x_2^{(n)})$$

positioned at the boundary of the unit disc.

Each microphone measures the sound it receives at $p$ different amount of time, and the sound collected by the $l$-th microphone at time $t_i$ is named $b_l(t_i)$.

Consider the data matrix $X$ such that $X_{i,j} = b_i(t_j)$, of shape $n \times p$. We will suppose that the sound measured by each microphone is corrupted by noise.

Both the data matrix $X$ of dimension $n \times p$ and the true sound source (in our example, there are $m = 4$ sound sources), are collected into the file `SoundSourceData.mat`.

1. Import the needed libraries (remember the function `scipy.io.loadmat` to load `.mat` files).
2. Use the functions of scipy.io to load the content of `SoundSourceData.mat`, and use it to identify the dataset $X$ and the true sound source matrix $F$. Use those matrix to find the values for $n$ and $p$ in this case (remember that $n \times p$ is the shape of $X$).
3. To solve convergence problems, set equals to 0 the values contained in $X$ which are strictly negative (those values exists because of the presence of noise).
4. Import the utility function NMF contained into the folder `./utils/NMF.py` and use it to compute the Non-Negative Matrix Factorization for the matrix $X$. Try different values for the number of iterations. Remember that the NMF of a positive matrix $X \in \mathbb{R}_+^{n \times p}$ is the matrix decomposition

$$X = WH \qquad \text{where } W \in \mathbb{R}_+^{n \times m}, H \in \mathbb{R}_+^{m \times p}$$

that minimizes the error measured in Frobenius norm

$$||X - WH||_F$$

among the set of positive matrices.

Below you will find the documentation on how to use the utility function `NMF` of `NMF.py`.

```
The NMF function takes as input:

X: the data matrix.
m: the number of source (known in advance).
T (optional): the number of iterations of the algorithm. The Default value is T=1000
tau (optional): the stopping criterion. The Default value is tau=1e-2.
return_error (optional): a boolean variable. if True, also returns the error vector.
                         The Default value is False.

And returns:

(W, H): tuple containing the NMF decomposition of X,
        where the shape of W is n x m, while the shape of H is m x p.
err (only if return_error = True): an array of length T (the number of iterations)
        that contains the behavior of the error during the iterations.
```

5. Using a subplot, show a $2 \times 4$ table of plots, where in the first line the rows of $F$ are shown (the true sound sources), while in the second one, you show the rows of $H$ (the approximated sound sources).
6. Comment the obtained results.