Dictionary Learning

Alain Rakotomamonjy
alain.rakoto@insa-rouen.fr,

Practical session description

This practical session aims at showing how dictionary learning can be used for solving a very simple source separation problem. We have an audio file containing three notes of piano and our objective is to retrieve 3 signals each related to one note. The main idea is to solve a non-negative dictionary learning, denoted as non-negative matrix factorization

$$\min_{\mathbf{W}\geq 0,\mathbf{H}\geq 0} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2$$

where $\mathbf{X}$ is a non-negative time-frequencey spectogram of the signal to decompose. By solving this problem, we hope to retrieve in $\mathbf{W}$ some frequency basis functions and in $\mathbf{H}$ their activations occuring the time.

Ex. 1 — Non-negative matrix factorization

1. Load the signal that we want to decompose and listen to it

   ```
   [x fs] = wavread('Mary');
   sound(x,fs);
   ```

2. compute its spectogram.

   ```
   FFTSIZE=1024;
   [V,Phi] = computeSpectrogram(x,fs);
   ```

   Here V is the time-frequency of the signal and Phi represents the phase of the signal at each time-frequency bin

   a) Plot the obtained time-frequency representation

   ```
   F = size(V,1);
   T = size(V,2);
   imagesc(db(V))
   set(gca,'YDir','normal')
   set(gca, 'XTickLabelMode', 'manual', 'XTickLabel', []);
   set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
   title('Spectrogram of Mary Had a Little Lamb')
   ylabel('Frequency')
   xlabel('Time')
   ```

   can you figure out the 3 different notes present in the signal?

   b) implement a proximal gradient descent algorithm that solves the non-negative dictionary learning problem. The implementation we propose is based on a coordinate descent algorithm which optimize alternatively $\mathbf{W}$ and $\mathbf{H}$.

   ```

   K=3;
   nbiter=200;
   [dim,nbsig]=size(V);
   W=1 + rand(dim,K);
   H=1 + rand(K,nbsig);

   for i=1:nbiter
       pas=1/norm(W'*W);
       for j=1:20
           H=H + pas*W'*(V-W*H);
           H= H.*(H>0);
       end;
   ```

```
    pas=1/norm(H*H');
    for j=1:20
        W=W+ pas*(V-W*H)*H';
        W=W.*(W>0);
    end;
end;
```

c) Plot the obtained basis functions

```
freq = linspace(0, fs/2, FFTSIZE/2+1);
time = linspace(0, length(x)/fs, T);
figure;
for i=1:K
    plot((i-1)*max(max(W))+(1-W(:,i)),freq,'LineWidth', 3)
    hold on
end
title('Basis Vectors')
ylabel('Frequency (Hz)')
xlabel('Basis')
set(gca, 'XTickLabelMode', 'manual', 'XTickLabel', []);
```

d) Plot the activation functions

```
figure;
for i=1:K
    plot(time, (i-1)*max(max(H))+(H(i,:)),'LineWidth', 3)
    hold on
end
ylabel('Activations')
xlabel('Time (seconds)')
set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
```

e) What happen to these basis functions and activations if we change the number of sources K.

3. Reconstructing the sources
   a) do the inverse spectrogram transform of each source and listen to them

```
[ xhat] = estimateSources(W,H,Phi);
 sound(xhat(:,1),fs)
 sound(xhat(:,2),fs)
 sound(xhat(:,3),fs)
```

   b) are you happy with this separation?

4. We want to improve our source separation by integrating in the learning problem some other constraints. For instance, we want the basis function norm to be smaller than 10 and the activation sources to be sparse, in order to remove some noises. The problem we want to solve is then

$$\min_{\mathbf{W}\geq 0,\mathbf{H}\geq 0} \quad \begin{array}{l} \|\mathbf{X}-\mathbf{WH}\|_F^2 + \sum_{i,j} H_{i,j} \\ \|\mathbf{W}_{.,j}\|_2 \leq 10 \end{array}$$

   a) For solving this problem, we need to add the proximal operator of the sparse positive constraint and the ℓ_2-norm ball constraint.

```
lambda=100;
W=1+rand(dim,K);
H=1+rand(K,nbsig);

for i=1:nbiter
    pas=1/norm(W'*W);
    for j=1:100
        H=H + pas*W'*(V-W*H);
        H=sign(H).*max(abs(H)-pas*lambda,0);
        H= H.*(H>0);
    end;
    pas=1/norm(H*H');
    for j=1:100
        W=W+ pas*(V-W*H)*H';
        W=W.*(W>0);
        normw=sqrt(sum(W.^2));
        for ii=1:K
        if normw(ii)>10
            W(:,ii)=W(:,ii)/normw(ii)*10;
        end;
        end;
    end;

end;
```

b) By looking at the obtained activation sources, select the parameter of the Lasso, you are happiest with.

```
figure;
for i=1:K
    plot(time, (i-1)*max(max(H))+(H(i,:)),'LineWidth', 3)
    hold on
end
ylabel('Activations')
xlabel('Time (seconds)')
set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
```

c) estimate the sources and listen to them.

```
[ xhat] = estimateSources(W,H,Phi);
 sound(xhat(:,1),fs)
 sound(xhat(:,2),fs)
 sound(xhat(:,3),fs)
```