Table of Contents

Supervised NMF Test Script

```
% This script performs supervised or semi-supervised NMF audio source
% separation. Modified from "Single-Channel Source Separation Tutorial
% Mini-Series" by Nicholas Bryan, Dennis Sun, and Eunjoon Cho

% https://ccrma.stanford.edu/~njb/teaching/sstutorial/
```

NOISY SPEECH INITIALIZATION

```
clear; close all; clc;

% Path to speech corpus files
path = 'path/to/speech/corpus';

% Noise source and talker gender of mixture to be separated
noise = 'SSN';      % Speech Shaped Noise
gender = 'MALE';    % Male Talker

% Noisey Speech Mixture
[xm, Fs] = audioread(['MIX_' gender '_' noise '.wav']);


%
 ========================================================================
% SPEECH TRAINING DATA. Speech training used was randomly selected
 from a
% corpus of the same gender but different talker as the target speech.
%
 ========================================================================

x1 = [];

% Female training samples
if strcmp(gender,'FEMALE')

    files = dir([path 'Stimuli - Speech/IEEE_NU/speakers/NCF014/audio/
*.wav']);
    randFile = randperm(length(files),30);

    % Concatenate random 30 speech examples to use as speech training
 data
    for n = 1:30
```

```
        in = audioread([files(randFile(n)).folder '/'
 files(randFile(n)).name]);
        x1 = [x1 ; in];

    end

% Male speech
elseif strcmp(gender,'MALE')

    files = dir([path 'Stimuli - Speech/IEEE_NU/speakers/NCM012/audio/
*.wav']);
    randFile = randperm(length(files),30);

    for n = 1:30

        in = audioread([files(randFile(n)).folder '/'
 files(randFile(n)).name]);
        x1 = [x1 ; in];

    end
end


%
 ========================================================================
% NOISE TRAINING DATA
%
 ========================================================================
if strcmp(noise,'SSN')
    x2 = audioread('speechShapedNoise.wav');

elseif strcmp(noise,'CONV')
    x2 = audioread('CST_Babble_Mono.wav');

end

% Ensure training data is normalized to full scale
x1 = x1 / max(abs(x1));
x2 = x2 / max(abs(x2));
```

STFT

```
FFTSIZE = 2048;
HOPSIZE = FFTSIZE / 2;
WINDOWSIZE = FFTSIZE;

% Spectrogram and magnitude of speech training data
X1 = myspectrogram(x1,FFTSIZE,Fs,hann(WINDOWSIZE),-HOPSIZE);
V1 = abs(X1(1:(FFTSIZE/2+1),:));

% Spectrogram and magnitude of noise training data
X2 = myspectrogram(x2,FFTSIZE,Fs,hann(WINDOWSIZE),-HOPSIZE);
```

```
V2 = abs(X2(1:(FFTSIZE/2+1),:));

% Spectrogram and magnitude of mixture
Xm = myspectrogram(xm,FFTSIZE,Fs,hann(WINDOWSIZE),-HOPSIZE);
Vm = abs(Xm(1:(FFTSIZE/2+1),:)); maxV = max(max(db(Vm)));

F = size(Vm,1);
T = size(Vm,2);

% Plot spectrograms
figure;
subplot(3,1,1)
imagesc(db(V1))
set(gca,'YDir','normal')
set(gca, 'XTickLabelMode', 'manual', 'XTickLabel', []);
set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
title('Speech')
ylabel('Frequency')
xlabel('Time')

subplot(3,1,2)
imagesc(db(V2))
set(gca,'YDir','normal')
set(gca, 'XTickLabelMode', 'manual', 'XTickLabel', []);
set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
title('Noise')
ylabel('Frequency')
xlabel('Time')

subplot(3,1,3)
imagesc(db(Vm))
set(gca,'YDir','normal')
set(gca, 'XTickLabelMode', 'manual', 'XTickLabel', []);
set(gca, 'YTickLabelMode', 'manual', 'YTickLabel', []);
title('speech-Noise Mixture')
ylabel('Frequency')
xlabel('Time')
```

NMF

```
%
 ========================================================================
% NMF PARAMETERS
%
 ========================================================================
K = [500 500];      % number of basis vectors
supervised = [1 1]; % binary vector specifying which sources are
 supervised
MAXITER = 100;      % total number of iterations to run
beta = 'KL';        % Divergence function ('IS', 'KL', or 'EU')

% Check if first source is supervised. If yes, perform NMF on training
 data
```

```
% for first source. Pass empty W, fixedInds
if supervised(1)
    [W1, H1] = nmfSS(V1, K(1), [], MAXITER, beta, []);
else
    W1 = 1+rand(F, K(1));
end

% Check if second source is supervised
if supervised(2)
    [W2, H2] = nmfSS(V2, K(2), [], MAXITER, beta, []);
else
    W2 = 1+rand(F, K(2));
end

% Set the fixed indices to the W matrices that are supervised
if supervised(1) && supervised(2)
 fixedInds = 1:sum(K);
elseif supervised(1)
    fixedInds = 1:K(1);
elseif supervised(2)
    fixedInds = (K(1)+1):sum(K);
else
    fixedInds = [];
end

% Perform separation
[W, H]   = nmfSS(Vm, K, [W1 W2], MAXITER, beta, fixedInds);
```

RECONSTRUCTION

```
% Save the mixture phase
phi = angle(Xm);
c = [1 cumsum(K)]; % Index variable for segmented portions of W (c =
 [1 500 1000] when K = [500 500]
for i=1:length(K)

    % Create masking filter for each source (first source first time
    % through loop, second source second time through loop, etc.)
    % Mask = (W_i * H_i) / (W * H)
    Mask =  W(:,c(i):c(i+1))*H(c(i):c(i+1),:)./(W*H);

    % Apply masking filter to original mixture
    XmagHat = Vm.*Mask;

    % Create upper half of frequency spectrum before ifft
    XmagHat = [XmagHat; conj( XmagHat(end-1:-1:2,:))];

    % Multiply with phase
    XHat = XmagHat.*exp(1i*phi);

    % convert source to time domain
    xhat(:,i) = real(invmyspectrogram(XmagHat.*exp(1i*phi),HOPSIZE))';
```

```
    % Normalize to full scale
    xhat(:,i) = xhat(:,i) / max(abs(xhat(:,i)));

end
```

Published with MATLAB® R2019b