Test of Lee & Seung algorithm for non-negative factorization S = CF
by Bill Menke for P. Pollisar, November 5, 2014

Reference: Lee & Seung, Algorithms for non-negative matrix factorization
http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf

My notation: **S** = **C F**; Lee and Seung's notation: **V** = **W H**

Based on iterative refinement of starting model, with either of these measures of the difference between $\mathbf{S}^{true}$ and $\mathbf{S}^{est}$:

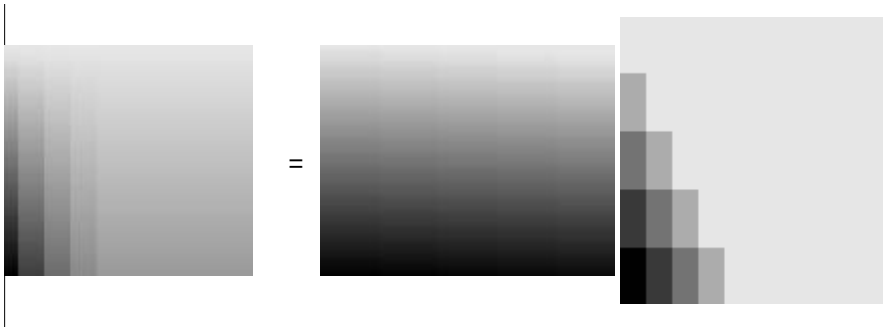
Distance: $R = \sum_i^N \sum_j^M \left(S_{ij}^{true} - S_{ij}^{obs}\right)^2$

Divergence: $D = \sum_i^N \sum_j^M \left(S_{ij}^{true} \ln \left(S_{ij}^{true} / S_{ij}^{obs}\right) + S_{ij}^{true} - S_{ij}^{obs}\right)$

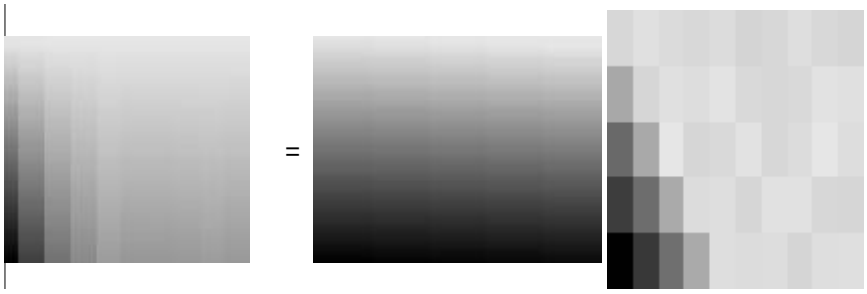
Procedure: The Matlab script LeeSeung2.m implements simple a $\mathbf{C}^{true}$, $\mathbf{F}^{true}$ and defines $\mathbf{S}^{true} = \mathbf{C}^{true}\,\mathbf{F}^{true}$ and then creates a starting model $\mathbf{C}^{(0)}\,\mathbf{F}^{(0)}$ by a smallish random perturbation of $\mathbf{C}^{true}$, $\mathbf{F}^{true}$. The script then improve this starting model by minimize either R or D using the Lee & Seung algorithm.

Results: Many runs give improvement in D and/or R of 95% or better after 200 iterations, so the algorithm achieves a good approximation to $\mathbf{S}^{true} = \mathbf{S}^{est}$. However, while final **C** and **F** always have non-negative elements, they do not always resemble $\mathbf{C}^{true}$ and $\mathbf{F}^{true}$, presumably because of the non-uniqueness inherent in the process.

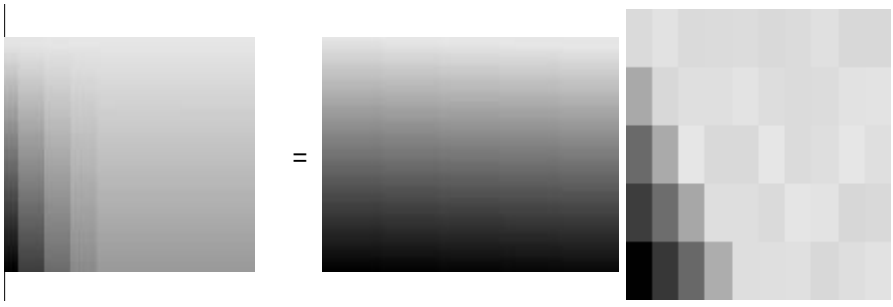
>> LeeSeung2
S is 100 by 10 with 5 factors
Ctrue has 0 negative values
Ftrue has 0 negative values
Strue has 0 negative values



distance minimization
initial C0 has 0 negative values
initial F0 has 0 negative values
initial S0 has 0 negative values
distance of S0 from Strue: 84963



200 interations
final distance reduction: **99.01** percent
final C0 has 0 negative values
final F0 has 0 negative values
final S0 has 0 negative values



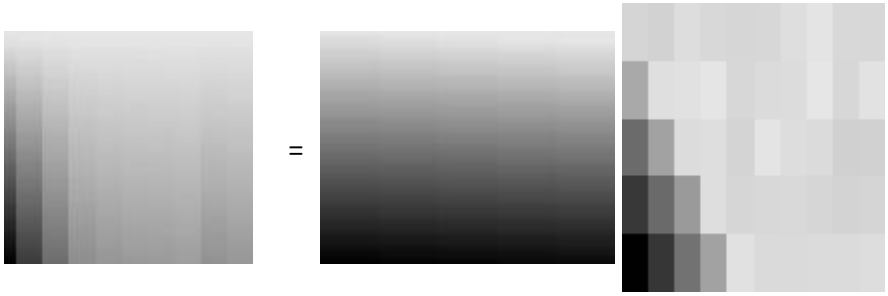
divergence minimization

initial C0 has 0 negative values

initial F0 has 0 negative values

initial S0 has 0 negative values

Divergence Strue||S0: 228



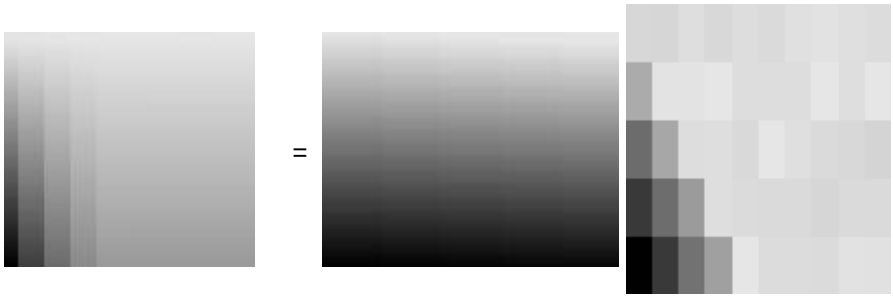
200 interations

final divergece reduction: **98.56** percent

final C0 has 0 negative values

final F0 has 0 negative values

final S0 has 0 negative values



```
clear all;

% Algorithm from Lee & Seung, Algorithms for non-negative
% matrix factorization

% factorization S = C F (my terminology)
% paper uses    V = W H

% true factorization
N=100;
M=10;
P=5;
fprintf('S is %d by %d with %d factors\n', N, M, P);
Ftrue = toeplitz([1:P]',[1,ones(1,M-1)]);
Ctrue = toeplitz([1:N]',[1,ones(1,P-1)]);
Strue = Ctrue * Ftrue;
figure(1);
eda_draw(Strue,' ','=',' ',' ',Ctrue,' ',Ftrue);

k = length(find(Ctrue<0));
fprintf('Ctrue has %d negative values\n', k );
k = length(find(Ftrue<0));
fprintf('Ftrue has %d negative values\n', k );
k = length(find(Strue<0));
fprintf('Strue has %d negative values\n', k );

fprintf('\n');
fprintf('distance minimization\n');

% perturbed factors
sigmaF = 0.1;
F0 = Ftrue-random('Normal',0,sigmaF,P,M);
F0(find(F0<0))=0;
sigmaC = 0.1;
C0 = Ctrue-random('Normal',0,sigmaC,N,P);
C0(find(C0<0))=0;
S0 = C0 * F0;
k = length(find(C0<0));
fprintf('initial C0 has %d negative values\n', k );
k = length(find(F0<0));
fprintf('initial F0 has %d negative values\n', k );
k = length(find(S0<0));
fprintf('initial S0 has %d negative values\n', k );
figure(2);
eda_draw(S0,' ','=',' ',' ',C0,' ',F0);

% Euclidean distance of true and estimated sample matrix
e = Strue(:)-S0(:);
E = e'*e;
E0 = E;
fprintf('distance of S0 from Strue: %.0f\n', E );
```

```
Niter=200;
fprintf('%d interations\n', Niter);
for iter=[1:Niter]

    M1num = C0'*Strue;
    M1den = C0'*C0*F0;
    M2num = Strue*F0';
    M2den = C0*F0*F0';

    for i=[1:P]
    for j=[1:M]
        F0(i,j) = F0(i,j)*M1num(i,j)/M1den(i,j);
    end
    end

    for i=[1:N]
    for j=[1:P]
        C0(i,j) = C0(i,j)*M2num(i,j)/M2den(i,j);
    end
    end
end

S0 = C0*F0;
e = Strue(:)-S0(:);
E = e'*e;
fprintf('final distance reduction: %.2f percent\n', 100-100*E/E0 );
k = length(find(C0<0));
fprintf('final C0 has %d negative values\n', k );
k = length(find(F0<0));
fprintf('final F0 has %d negative values\n', k );
k = length(find(S0<0));
fprintf('final S0 has %d negative values\n', k );
fprintf('\n');
figure(3);
eda_draw(S0,' ','=',' ',' ',C0,' ',F0);

fprintf('divergence minimization\n');

% perturbed factors
sigmaF = 0.1;
F0 = Ftrue-random('Normal',0,sigmaF,P,M);
F0(find(F0<0))=0;
sigmaC = 0.1;
C0 = Ctrue-random('Normal',0,sigmaC,N,P);
C0(find(C0<0))=0;
S0 = C0 * F0;
k = length(find(C0<0));
fprintf('initial C0 has %d negative values\n', k );
k = length(find(F0<0));
fprintf('initial F0 has %d negative values\n', k );
k = length(find(S0<0));
fprintf('initial S0 has %d negative values\n', k );
```

```
figure(4);
eda_draw(S0,' ','=',' ',' ',C0,' ',F0);

% Divergence of true and estimated sample matrix
D = 0;
for i=[1:N]
for j=[1:M]
    D = D + Strue(i,j)*log(Strue(i,j)/S0(i,j)) - Strue(i,j) + S0(i,j);
end
end
D0 = D;
fprintf('Divergence Strue||S0: %.0f\n', D );

Niter=200;
fprintf('%d interations\n', Niter);
for iter=[1:Niter]

    % S(N,M) C(N,P) F(P,M)
    for i=[1:P]
    for j=[1:M]
        num = 0;
        den = 0;
        for k=[1:N]
            num = num + C0(k,i)*Strue(k,j)/S0(k,j);
            den = den + C0(k,i);
        end
        F0(i,j) = F0(i,j)*num/den;
    end
    end

    % S(N,M) C(N,P) F(P,M)
    for i=[1:N]
    for j=[1:P]
        num = 0;
        den = 0;
        for k=[1:M]
            num = num + F0(j,k)*Strue(i,k)/S0(i,k);
            den = den + F0(j,k);
        end
        C0(i,j) = C0(i,j)*num/den;
    end
    end

S0 = C0*F0;
end


% Divergence of true and estimated sample matrix
D = 0;
for i=[1:N]
for j=[1:M]
    D = D + Strue(i,j)*log(Strue(i,j)/S0(i,j)) - Strue(i,j) + S0(i,j);
```

```
end
end
fprintf('final divergece reduction: %.2f percent\n', 100-100*D/D0 );
k = length(find(C0<0));
fprintf('final C0 has %d negative values\n', k );
k = length(find(F0<0));
fprintf('final F0 has %d negative values\n', k );
k = length(find(S0<0));
fprintf('final S0 has %d negative values\n', k );
figure(5);
eda_draw(S0,' ','=',' ',' ',C0,' ',F0);
```