Homework Set 2, CPSC 8420, Fall 2020

Last Name, First Name

Due 10/15/2020, Thursday, 11:59PM EST

Problem 1

Consider Non-negative Matrix Factorization Problem: $\underset{\mathbf{A},\ \mathbf{G}\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{X}-\mathbf{AG}\|_F^2$.

1. Please show that by leveraging gradient descent method with specific element-wise learning rate in each iteration, we will have:

$$\begin{aligned}\mathbf{A}_{k+1} &= \mathbf{A}_k \odot \frac{\mathbf{X}\mathbf{G}_k^T}{\mathbf{A}_k\mathbf{G}_k\mathbf{G}_k^T}\\ \mathbf{G}_{k+1} &= \mathbf{G}_k \odot \frac{\mathbf{A}_{k+1}^T\mathbf{X}}{\mathbf{A}_{k+1}^T\mathbf{A}_{k+1}\mathbf{G}_k}\end{aligned} \tag{1}$$

   where we use $\mathbf{Z}\odot\mathbf{Y}$, $\frac{\mathbf{Z}}{\mathbf{Y}}$ to represent component-wise multiplication and division between the corresponding elements of $\mathbf{Z}$ and $\mathbf{Y}$ respectively.

2. We say $\mathbf{A}$ is the feature matrix, please use the given *atnt* dataset to plot the 10 feature faces by using *MUA* to obtain $\mathbf{A}$ after 500 iterations with random initializations $(\mathbf{A}_0, \mathbf{G}_0)$.

1. Denote $\mathbf{J} = \frac{1}{2}\|\mathbf{X}-\mathbf{AG}\|_F^2$, take the derivative w.r.t. $\mathbf{A}, \mathbf{G}$ respectively, we have: $\nabla_{\mathbf{A}}\mathbf{J} = (\mathbf{AG}-\mathbf{X})\mathbf{G}^T, \nabla_{\mathbf{G}}\mathbf{J} = \mathbf{A}^T(\mathbf{AG}-\mathbf{X})$, then according to GD:

$$\begin{aligned}\mathbf{A}_{k+1} &= \mathbf{A}_k - \eta_1\nabla_{\mathbf{A}}\mathbf{J} = \mathbf{A}_k - \eta_1(\mathbf{A}_k\mathbf{G}_k-\mathbf{X})\mathbf{G}_k^T\\ \mathbf{G}_{k+1} &= \mathbf{G}_k - \eta_2\nabla_{\mathbf{G}}\mathbf{J} = \mathbf{G}_k - \eta_2\mathbf{A}_{k+1}^T(\mathbf{A}_{k+1}\mathbf{G}_k-\mathbf{X})\end{aligned} \tag{2}$$

if we set $\eta_1 = \frac{\mathbf{A}_k}{\mathbf{A}_k\mathbf{G}_k\mathbf{G}_k^T}$ $\eta_2 = \frac{\mathbf{G}_k}{\mathbf{A}_{k+1}^T\mathbf{A}_{k+1}\mathbf{G}_k}$, we will obtain the updating algorithm in Eq. (1).

2. We can use the following codes:

```
clear all; close all;
load('atnt40x10x112x92ML5FoldNaturalCrossValidationSplit.mat')
data=M';
rank=10;
[row,col]=size(data);
A = 256*rand(row,rank);
G = rand(rank,col);
```

```
for j=1:500
  A = A.*((data*G')./(A*G*G'));
  G = G.*((A'*data)./(A'*A*G));
end
count=0;
figure
for k=1:2
    for t=1:5
        count=count+1;
        subplot(2,5,count)
        imshow(reshape(A(:,count),112,92),[])
    end
end
```

Problem 2

Please propose an alternating minimization method to update $\mathbf{A}$ and $\mathbf{G}$ column-by-column with close solution: $\underset{\mathbf{A},\ \mathbf{G}\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{X}-\mathbf{A}\mathbf{G}^T\|_F^2$. Then write a program to demonstrate that the objective is monotonically non-increasing with updates (certain figure is expected).

By noticing $\mathbf{A}\mathbf{G}^T=\sum_{i=1}^p \mathbf{a}_i\mathbf{g}_i^T$, *where* $\mathbf{a}_i$ *and* $\mathbf{g}_i$ *denote the* i*-th column of* $\mathbf{A},\mathbf{G}$ *respectively, we can reformulate the objective as* $\underset{\mathbf{a}_j,\ \mathbf{g}_j\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{X}-\sum_{i\neq j}^p \mathbf{a}_i\mathbf{g}_i^T-\mathbf{a}_j\mathbf{g}_j^T\|_F^2$. *Since we utilize alternating minimization to update* $\mathbf{a}_j,\ \mathbf{g}_j$ *while fixing the rest, we can regard* $\tilde{\mathbf{X}}=\mathbf{X}-\sum_{i\neq j}^p \mathbf{a}_i\mathbf{g}_i^T$ *as constant, we reformulate the equation as:* $\underset{\mathbf{a},\ \mathbf{g}\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\|\tilde{\mathbf{X}}-\mathbf{a}\mathbf{g}^T\|_F^2=\underset{\mathbf{a},\ \mathbf{g}\geq\mathbf{0}}{\text{minimize}}\ \frac{1}{2}\,\boldsymbol{tr}\left(\mathbf{a}\mathbf{g}^T\mathbf{g}\mathbf{a}^T-2\tilde{\mathbf{X}}\mathbf{g}\mathbf{a}^T\right)$, *therefore we have the following updating algorithm:*

$$\begin{aligned}\mathbf{a}^+&=\arg\min_{\mathbf{a}}\,\boldsymbol{tr}\left(\mathbf{a}\mathbf{g}^T\mathbf{g}\mathbf{a}^T-2\tilde{\mathbf{X}}\mathbf{g}\mathbf{a}^T\right)=\arg\min_{\mathbf{a}}\|\mathbf{a}-\frac{\tilde{\mathbf{X}}\mathbf{g}}{\|\mathbf{g}\|^2}\|^2=\max\{\frac{\tilde{\mathbf{X}}\mathbf{g}}{\|\mathbf{g}\|^2},0\}\\ \mathbf{g}^+&=\arg\min_{\mathbf{g}}\,\boldsymbol{tr}\left(\mathbf{g}\mathbf{a}^T\mathbf{a}\mathbf{g}^T-2\tilde{\mathbf{X}}^T\mathbf{a}\mathbf{g}^T\right)=\arg\min_{\mathbf{g}}\|\mathbf{g}-\frac{\tilde{\mathbf{X}}^T\mathbf{a}}{\|\mathbf{a}\|^2}\|^2=\max\{\frac{\tilde{\mathbf{X}}^T\mathbf{a}}{\|\mathbf{a}\|^2},0\}\end{aligned}\tag{3}$$

Algorithm 1 Alternating Linearized Minimization for MUA

Input: data $\mathbf{X} \in \mathbb{R}^{m\times n}$ and number of iterations K
Initialization: $\mathbf{A}_0 \in \mathbb{R}^{m\times p}, \mathbf{G}_0 \in \mathbb{R}^{n\times p}$
while $k \leq K$ **do**
  while $j \leq p$ **do**
    optimize j-th column of $\mathbf{A}_{k+1}$ and $\mathbf{G}_{k+1}$ by Eq. (3)
  end while
end while
Output: $\mathbf{A}_K$ and $\mathbf{G}_K$

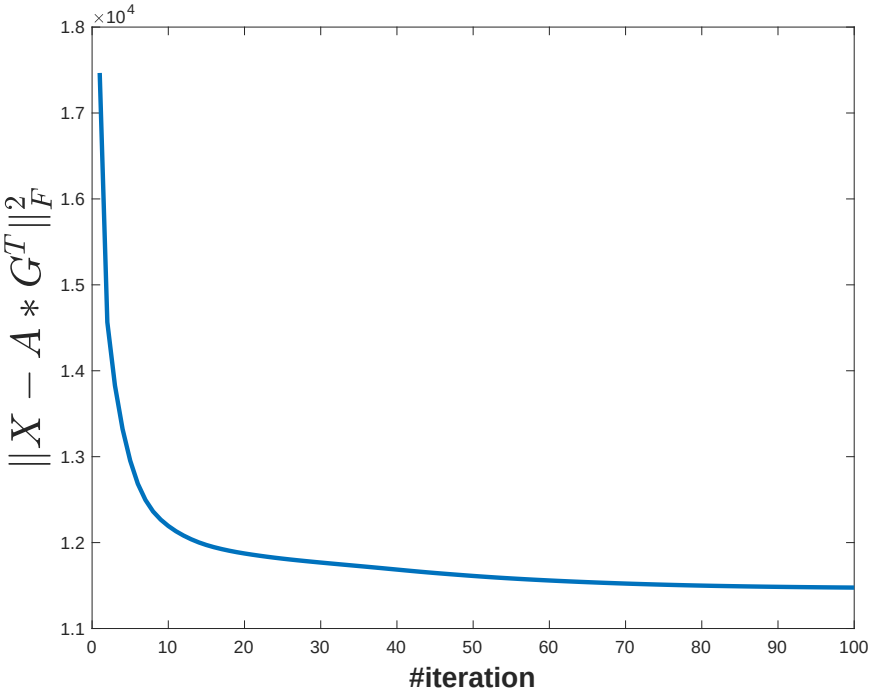
```
m=50;
n=40;
p=20;
X=10*rand(m,n);
A=10*rand(m,p);
G=rand(n,p);
loss=zeros(100,1);
for k=1:100
    for kk=1:p
        X_bar=X;
        for kkk=1:p
            if kkk ~=kk
                X_bar=X_bar-A(:,kkk)*G(:,kkk)';
            end
        end
        A(:,kk)=max(X_bar*G(:,kk)/(norm(G(:,kk))^2),0);
        G(:,kk)=max(X_bar'*A(:,kk)/(norm(A(:,kk))^2),0);
    end
    loss(k)=norm(X-A*G','fro')^2;
end
plot(loss,'LineWidth',2.5,'MarkerSize',20)
xlabel('#iteration','FontSize',16,'FontWeight','bold')
ylabel('$\|X-A*G^T\|^2_F$','interpreter','latex','FontSize',24,'FontWeight','bold')
```

Problem 3

Let's revisit Least Squares Problem: $\underset{\boldsymbol{\beta}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{y}-\mathbf{A}\boldsymbol{\beta}\|_2^2$, where $\mathbf{A} \in \mathbb{R}^{n\times p}$.

1. Please show that if $p > n$, then vanilla solution $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$ is not applicable any more. *Since $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{p\times p}$, on the other side $rank(\mathbf{A}^T\mathbf{A}) \leq \min\{n,p\} < p$, thus $\mathbf{A}^T\mathbf{A}$ is not invertable, and $(\mathbf{A}^T\mathbf{A})^{-1}$ operation will cause error.*

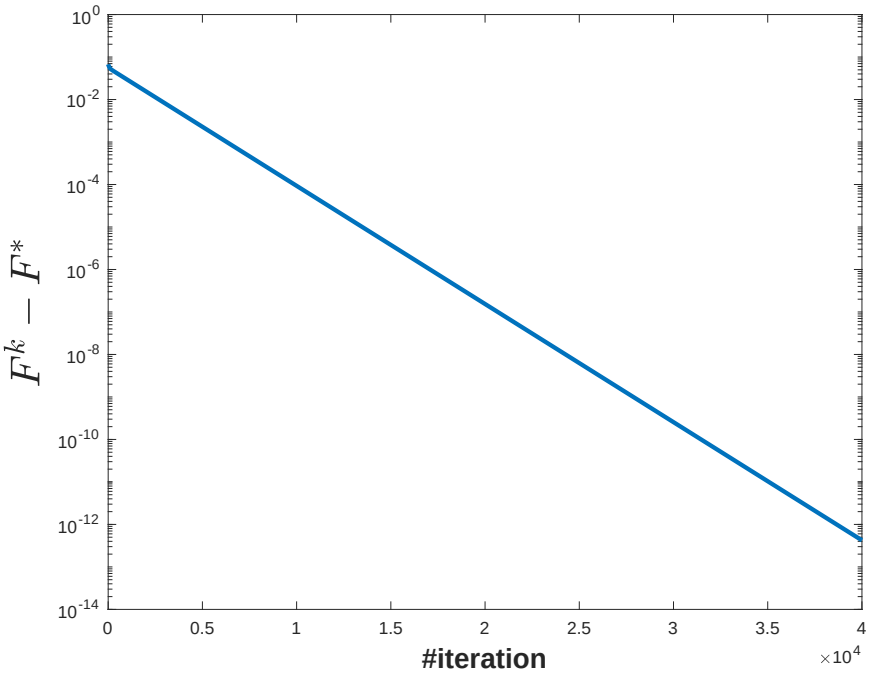
2. Let's assume $\mathbf{A} = [1,2,4;1,3,5;1,7,7;1,8,9], \mathbf{y} = [1;2;3;4]$. Please show via experiment results that Gradient Descent method will obtain the optimal solution with Linear Convergence rate if the learning rate is fixed to be $\frac{1}{\sigma_{max}(\mathbf{A}^T\mathbf{A})}$, and $\boldsymbol{\beta}_0 = [0;0;0]$.



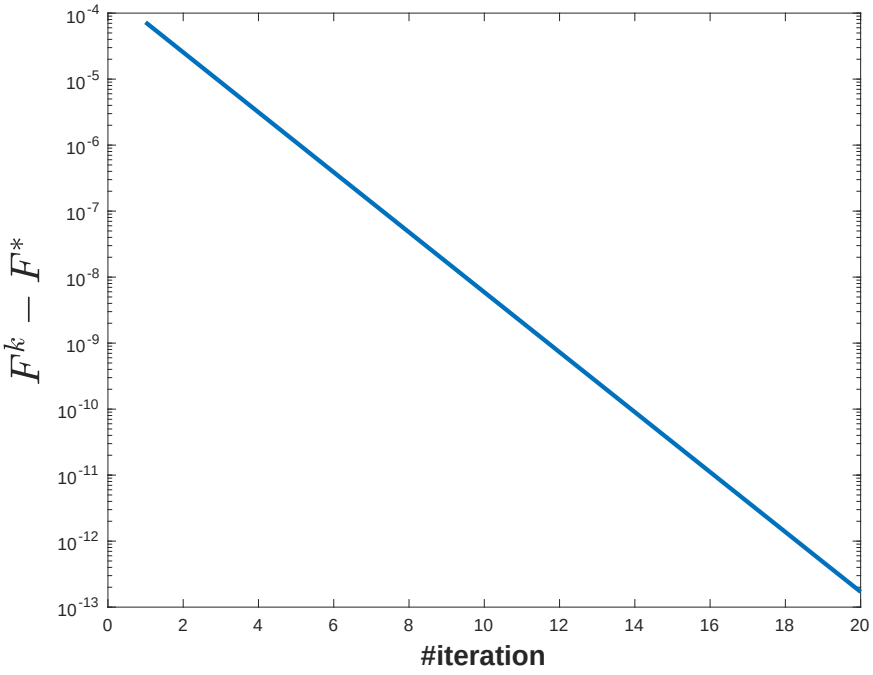

```
%set(gcf,'Position',[100 300 650 300]);
set(gca, 'LineWidth' , 1.5,'FontSize',6);
maxItr=40000;
gap=zeros(maxItr,1);
A=[1 2 4;1 3 5; 1 7 7; 1 8 9];
y=[1;2;3;4];
x_opt = (A'*A)\(A'*y);
obj_opt = 0.5*norm(y-A*x_opt)^2;
[U,S,V]=svd(A'*A);
L = S(1,1);
x = [0;0;0];
for i=1:maxItr
    x = x - 1/L*(A'*A*x-A'*y);
    gap(i)=0.5*norm(y-A*x)^2-obj_opt;
end
semilogy(1:maxItr,gap,'LineWidth',2.5,'MarkerSize',20)
xlabel('#iteration','FontSize',16,'FontWeight','bold')
ylabel('$F^k-F^*$','interpreter','latex','FontSize',24,'FontWeight','bold')
```

3. Now let's consider ridge regression: $\underset{\boldsymbol{\beta}}{\text{minimize}}\ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$, where $\mathbf{A}, \mathbf{y}, \boldsymbol{\beta}_0$ remains the same as above while learning rate is fixed to be $\frac{1}{\lambda+\sigma_{max}(\mathbf{A}^T\mathbf{A})}$ where λ varies from $0.1, 1, 10, 100, 200$, please show that Gradient Descent method with larger λ converges faster. (You will receive up to 10 bonus points if you can prove faster convergence rate mathematically.)

```
%set(gcf,'Position',[100 300 650 300]);
set(gca, 'LineWidth' , 1.5,'FontSize',6);
```




```
maxItr=20;
gap=zeros(maxItr,1);
A=[1 2 4;1 3 5; 1 7 7; 1 8 9];
y=[1;2;3;4];
lambda = 200;
x_opt = (A'*A+lambda*eye(3))\(A'*y);
obj_opt = 0.5*norm(y-A*x_opt)^2+0.5*lambda*norm(x_opt)^2;
[U,S,V]=svd(A'*A);
L = S(1,1)+lambda;
x = [0;0;0];
for i=1:maxItr
    x = x - 1/L*((A'*A+lambda*eye(3))*x-A'*y);
    gap(i)=0.5*norm(y-A*x)^2+0.5*lambda*norm(x)^2-obj_opt;
end
semilogy(1:maxItr,gap,'LineWidth',2.5,'MarkerSize',20)
xlabel('#iteration','FontSize',16,'FontWeight','bold')
ylabel('$F^k-F^*$','interpreter','latex','FontSize',24,'FontWeight','bold')
```



Problem 4

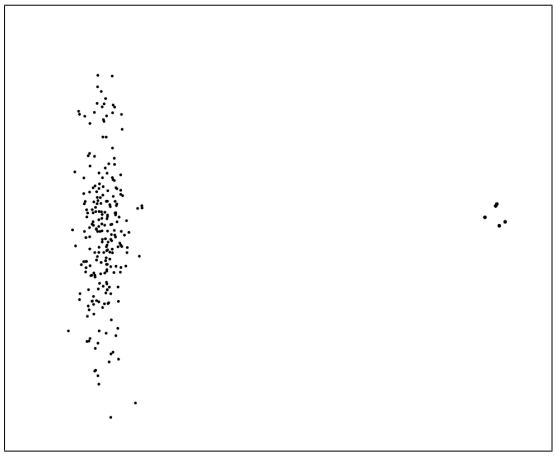
Please download the image from `https://en.wikipedia.org/wiki/Lenna#/media/File:Lenna_(test_image).png` with dimension $512 \times 512 \times 3$. Assume for each RGB channel data X, we have $[U, \Sigma, V] = svd(X)$. Please show each compression ratio and reconstruction image if we choose first $2, 5, 20, 50, 80, 100$ components respectively. Also please determine the best component number to obtain a good trade-off between data compression ratio and reconstruction image quality. (Open question, that is your solution will be accepted as long as it's reasonable.)

If we remain the first 50 components, then the compression ratio is around $\frac{50*512+50*50+50*512}{512*512} \approx 0.2$

```
close all;
numberOfComponents = 50;
lenna_rec = zeros(size(Lenna));
for channel=1:3
    a = Lenna(:,:,channel);
    a = double(a);
    [U,S,V]=svd(a);
    U_s = U(:,1:numberOfComponents);
    S_s = S(1:numberOfComponents,1:numberOfComponents);
    V_s = V(:,1:numberOfComponents);
    lenna_rec(:,:,channel)=U_s*S_s*V_s';
end
imshow(uint8(lenna_rec))
```

Problem 5

Draw the first two principle components for the following dataset. Based on this, comment on a



limitation of PCA and propose a possible algorithm/method to improve it.

One of the limitations of PCA is that is can be highly affected by just a few outliers. Methods include but not limited to detect and remove outlier, use a robust correlation matrix, or formulate robust norm such as ℓ_1 or $\ell_{2,1}$ objective function, etc.