CMSE 820: Homework #5

Due on October 20, 2019 at 11:59pm

Professor Yuying Xie

Boyao Zhu

Problem 1

Solution
a.
The $l_{2,1}$ norm

$$f(X) = \|X\|_{2,1} = \Sigma_j \|X_{\cdot,j}\|_2 = \Sigma_j \sqrt{\Sigma_i X_{ij}^2}$$

To show that $f(X)$ is a convex function X, one needs to show that $f(X)$ is a norm of X, as convexity follows directly from the definition of norms.

- By construction, it's clear that for any matrix $X, f(X) \geq 0$.
- $$f(X) = 0 \Leftrightarrow \Sigma_j \sqrt{\Sigma_i X_{ij}^2} = 0 \Leftrightarrow \forall i, j, X_{ij} = 0 \Leftrightarrow X = 0$$
- $\forall \alpha \in \mathbb{R}$
$$f(\alpha X) = \Sigma_j \sqrt{\Sigma_i (\alpha X_{ij})^2} = |x| \Sigma_j \sqrt{\Sigma_i X_{ij}^2} = |\alpha| f(X)$$
- $\forall X, Y$
$$f(X+Y) = \Sigma_j \|(X+Y)_{\cdot,j}\|_2 \leq \Sigma_j (\|X_{\cdot,j}\|_2 + \|Y_{\cdot,j}\|_2) = \Sigma_j \|X_{\cdot,j}\|_2 + \Sigma_j \|Y_{\cdot,j}\|_2 = f(X) + f(Y).$$
Note that the inequality follows from the triangle inequality for the l_2 norm of vectors.

b.

- If $X_{\cdot,j} \neq \mathbf{0}, f(X)$ is differentiable and convex, so
$$(\partial \|X\|_{2,1})_{ij} = \frac{\partial f(X)}{\partial X_{ij}} = \frac{X_{ij}}{\|X_{\cdot,j}\|_2}$$
- If $X_{\cdot,j} = \mathbf{0}, \text{and} \forall Y$ such that only the j-th column $Y_{\cdot,j}$ differs from $X_{\cdot,j}$.
$$f(Y) - f(X) = \|Y_{\cdot,j}\|_2$$

Therefore,

$$(\partial \|X\|_{2,1})_{ij}(X_{\cdot,j} = \mathbf{0}) = \{W_{ij} : \|W_{\cdot,j}\|_2 \leq 1\}$$

c.
The optimization problem

$$\min_A \frac{1}{2} \|X - A\|_F^2 + \tau \|A\|_{2,1}$$

The corresponding Lagragian function $\mathcal{L}(A) = \frac{1}{2}\|X - A\|_F^2 + \tau\|A\|_{2,1}$ with subgradient:

$$(\partial \mathcal{L})_{\cdot,j} = -(X_{\cdot,j} - A_{\cdot,j}) + \begin{cases} \tau \frac{A_{\cdot,j}}{\|A_{\cdot,j}\|_2}, & A_{\cdot,j} \neq \mathbf{0} \\ \tau W_{\cdot,j} : \|W_{\cdot,j}\|_2 \leq 1, & A_{\cdot,j} = \mathbf{0} \end{cases}$$

- If $\|X_{\cdot,j}\|_2 \geq \tau$, let $A_{\cdot,j} = (1 - \frac{\tau}{\|X_{\cdot,j}\|_2}) X_{\cdot,j}$, we can verify that
$$(\partial \mathcal{L})_{\cdot,j} = -(X_{\cdot,j} - (1 - \frac{\tau}{\|X_{\cdot,j}\|_2}) X_{\cdot,j}) + \tau \frac{X_{\cdot,j}}{\|X_{\cdot,j}\|_2} = \mathbf{0}$$

- If $\|X_{\cdot,j}\|_2 \leq \tau$, let $A_{\cdot,j} = \mathbf{0}$ and choose $W_{\cdot,j} = X_{\cdot,j}/\tau(\|W_{\cdot,j}\|_2 \leq 1)$, then

$$(\partial\mathcal{L}_{\cdot,j} = -(X_{\cdot,j}) + \tau\frac{X_{\cdot,j}}{\tau} = \mathbf{0}$$

Since $\|\cdot\|_F^2$ is strictly convex (note the power 2 in the subscript), the optimization problem is also strictly convex. The unique optimal solution is then given by

$$A = X\mathcal{S}_\tau(\text{diag}(\mathbf{x}))\text{diag}(\mathbf{x})^{-1}$$

where $x_j = \|X_{\cdot,j}\|_2$ and the j-th entry of $\text{diag}(\mathbf{x})^{-1}$ is zero if $x_j = 0$.

Problem 2

Solution

a.

Consider some arbitrary x, y and $t \in [0,1]$. By the definition of $f(x)$, there exist $a, b, c(t) \in S$ such that

$$f(x) = f_a(x), \quad f(y) + f_b(y), \quad f[tx + (1-t)y] = f_{c(t)}[tx + (1-t)y]$$

Since $f_c(t)$ are convex, we have

$$f[tx+(1-t)y] = f_{c(t)}[tx+(1-t)y] \leq tf_{c(t)}(x)+(1-t)f_{c(t)}(y) \leq tf_a(x)+(1-t)f_b(y) = tf(x)+(1-t)f(y)$$

Hence, $f(x)$ is convex.

b.

Let's consider the following general minimization problem:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & h_i(x) \leq 0, i = 1, \cdots, m, \\ & l_j(x) = 0, j = 1, \cdots, n. \end{aligned}$$

The corresponding Lagrangian reads

$$L(x, v, u) = f(x) + v^T h(x) + u^T l(x), \quad v \in \mathbb{R}^m, u \in \mathbb{R}^n$$

The dual function is

$$g(u, v) = \min_x L(x, u, v)$$

Now consider some arbitrary $u_1, u_2 \in \mathbb{R}^m, v_1, v_2 \in \mathbb{R}^n$ and $t \in [0,1]$, we have

$$\begin{aligned} g[tu_1 + (1-t)u_2, tv_1 + (1-t)v_2] &= \min_x\{f(x) + t[u_1^T h(x) + v_1^T l(x)] + (1-t)[u_2^T h(x) + v_2^T l(x)]\} \\ &\geq t\min_x[f(x) + u_1^T h(x) + v_1^T l(x)] + (1-t)\min_x[f(x) + u_2^T h(x) + v_2^T l(x)] \\ &= tg(u_1, v_1) + (1-t)g(u_2, v_2) \end{aligned}$$

which shows that the dual function $g(u, v)$ is concave.
The constraint $u \geq 0$ simply adds a linear term to the Lagrangian, which is both concave and convex. The dual problem

$$\underset{u\geq 0, v}{max} g(u, v) = -\underset{u\geq 0, v}{min} - g(u, v)$$

is thus a convex optimization problem.

Problem 3

The Lagrangian for the primal problem reads

$$L(x,u,v) = \frac{1}{2}x^TQx + c^Tx - u^Tx + v^T(Ax-b)$$

Note that $L(x,u,v)$ is a strictly convex function of x, because the quadratic term is strictly convex and the affine terms are convex. Thus, $\min\limits_x L(x,u,v)$ has a unique solution x^* satisfying $\frac{\partial L}{\partial x}|_{x=x^*}= 0$

$$\frac{\partial L}{\partial x}\Big|_{x=x^*} = Qx^* + c - u + A^Tv = 0 \Leftrightarrow x^* = -Q^{-1}(c-u+A^Tv)$$

Note also that since Q is positive definite, Ker$Q$=0, from which its invertibility follows.
The dual problem is

$$\max_{u\geq 0,v} g(u,v) + \max_{u\geq 0,v} L(x^*,u,v) = \max_{u\geq 0,v} -\frac{1}{2}(c-u+A^Tv)^TQ^{-1}(c-u+A^Tv) + v^Tb.$$

Problem 4

I attached coding in the back. But the program is very slow so that I cannot get the results on time before submitting. I will upload final results when my program is done.

Untitled7

October 20, 2019

[54]:
```python
def fun(X,omega,tau,step):
    a = np.zeros_like(X)
    z = np.zeros_like(X)
    znew = np.ones_like(X)
    count = 0
    while np.linalg.norm(z-znew)/np.linalg.norm(z)>1e-2 or count>5000:
        count += 1
        z = np.copy(znew)
        u,s,v = np.linalg.svd(z,full_matrices=False)
        s_tau = np.zeros_like(s)
        for i in range(s.size):
            if s[i]>tau:
                s_tau[i] = s[i]-tau
            elif s[i]<-tau:
                s_tau[i] = s[i]+tau
            else:
                s_tau[i] = 0
        a = np.dot(np.dot(u,np.diag(s_tau)),v)
        a_filter = np.copy(a)
        for (i,j) in omega:
            a_filter[i,j]=0
        znew = z+step*(X-a_filter)
    return a,z,znew
```

[ ]:
```python
import csv
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


X1 = pd.read_csv("M1.csv").as_matrix().reshape(-1).T
X2 = pd.read_csv("M2.csv").as_matrix().reshape(-1).T
X3 = pd.read_csv("M3.csv").as_matrix().reshape(-1).T
X4 = pd.read_csv("M4.csv").as_matrix().reshape(-1).T
X5 = pd.read_csv("M5.csv").as_matrix().reshape(-1).T
X6 = pd.read_csv("M6.csv").as_matrix().reshape(-1).T
X7 = pd.read_csv("M7.csv").as_matrix().reshape(-1).T
```

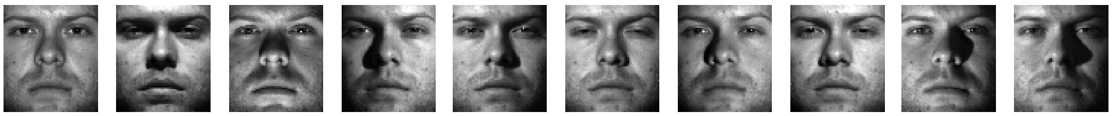
```python
X8 = pd.read_csv("M8.csv").as_matrix().reshape(-1).T
X9 = pd.read_csv("M9.csv").as_matrix().reshape(-1).T
X10= pd.read_csv("M10.csv").as_matrix().reshape(-1).T
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)

# total number of data 320880

from numpy import random

colplot,rowplot = 10,1
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(X[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 10% of missing
col = random.randint(0,9,10)
row = random.randint(0,32088,32088)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
error = []
A,Z,Z_new = fun(X,omega,1000,0.1)
error.append(np.linalg.norm(A-X))
print ("%10 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 20% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*2)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
```

```python
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%20 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()


# 30% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*3)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%30 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 40% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*4)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
```

```python
print ("%40 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()


plt.plot(error)
plt.show()
```

```
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  import sys
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.

/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  if __name__ == '__main__':
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  # Remove the CWD from sys.path while we load stuff.
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  # This is added back by InteractiveShellApp.init_path()
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  if sys.path[0] == '':
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:13:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  del sys.path[0]
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.

/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:15:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
```

```
  from ipykernel import kernelapp as app
/Users/boyaozhu/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:16:
FutureWarning: Method .as_matrix will be removed in a future version. Use
.values instead.
  app.launch_new_instance()
```



[ ]:
```
X1 = pd.read_csv("H1.csv").as_matrix().reshape(-1).T
X2 = pd.read_csv("H2.csv").as_matrix().reshape(-1).T
X3 = pd.read_csv("H3.csv").as_matrix().reshape(-1).T
X4 = pd.read_csv("H4.csv").as_matrix().reshape(-1).T
X5 = pd.read_csv("H5.csv").as_matrix().reshape(-1).T
X6 = pd.read_csv("H6.csv").as_matrix().reshape(-1).T
X7 = pd.read_csv("H7.csv").as_matrix().reshape(-1).T
X8 = pd.read_csv("H8.csv").as_matrix().reshape(-1).T
X9 = pd.read_csv("H9.csv").as_matrix().reshape(-1).T
X10= pd.read_csv("H10.csv").as_matrix().reshape(-1).T
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)

# total number of data 320880

from numpy import random

colplot,rowplot = 10,1
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(X[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 10% of missing
col = random.randint(0,9,10)
row = random.randint(0,32088,32088)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
error = []
```

```
A,Z,Z_new = fun(X,omega,1000,0.1)
error.append(np.linalg.norm(A-X))
print ("%10 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 20% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*2)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%20 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()


# 30% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*3)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%30 missing")
fig = plt.figure(figsize=(13,13))
```

```
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 40% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*4)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%40 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()


plt.plot(error)
plt.show()
```

[ ]:
```
X1 = pd.read_csv("F1.csv").as_matrix().reshape(-1).T
X2 = pd.read_csv("F2.csv").as_matrix().reshape(-1).T
X3 = pd.read_csv("F3.csv").as_matrix().reshape(-1).T
X4 = pd.read_csv("F4.csv").as_matrix().reshape(-1).T
X5 = pd.read_csv("F5.csv").as_matrix().reshape(-1).T
X6 = pd.read_csv("F6.csv").as_matrix().reshape(-1).T
X7 = pd.read_csv("F7.csv").as_matrix().reshape(-1).T
X8 = pd.read_csv("F8.csv").as_matrix().reshape(-1).T
X9 = pd.read_csv("F9.csv").as_matrix().reshape(-1).T
X10= pd.read_csv("F10.csv").as_matrix().reshape(-1).T
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)

# total number of data 320880

from numpy import random
```

```
colplot,rowplot = 10,1
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(X[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 10% of missing
col = random.randint(0,9,10)
row = random.randint(0,32088,32088)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
error = []
A,Z,Z_new = fun(X,omega,1000,0.1)
error.append(np.linalg.norm(A-X))
print ("%10 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 20% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*2)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%20 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
```

```
    plt.axis("off")
plt.show()


# 30% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*3)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%30 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()

# 40% of missing
X = np.array([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10]).T.astype(float)
col = random.randint(0,9,10)
row = random.randint(0,32088,32088*4)

omega = []
for i in row:
    for j in col:
        X[i,j]=0
        omega.append(np.array([i,j]))
omega = np.array(omega)
A,Z,Z_new = fun(X,omega,10000,0.1)
error.append(np.linalg.norm(A-X))
print ("%40 missing")
fig = plt.figure(figsize=(13,13))
for i in range(1,colplot*rowplot+1):
    fig.add_subplot(rowplot,colplot,i)
    plt.imshow(A[:,i-1].reshape(191,168),cmap="gray")
    plt.axis("off")
plt.show()
```

```
plt.plot(error)
plt.show()
```