EN.553.662: Optimization for Data Science
Homework 2: Gradient Descent

Ha Manh Bui (CS Department)
hbui13@jhu.edu

Spring 2023

1 Problem 1

Note: This question must be solved without invoking any result on the diagonalization of symmetric matrices.

Let A be an $n \times n$ symmetric matrix and $\Omega = \mathbb{R}^n \setminus \{0\}$. Define, for $x \in \Omega$, the function

$$F(x) = \frac{x^\top A x}{x^\top x}.$$

(1) Using the fact that $F(x) = F(x/|x|)$ for all $x \in \Omega$, prove that $\underset{\Omega}{\operatorname{argmin}} F$ and $\underset{\Omega}{\operatorname{argmax}} F$ are not empty.

Proof. We have

$$F(x) = \frac{x^\top A x}{x^\top x} = \frac{\sum_{i=1}^n \lambda_i y_i^2}{\sum_{i=1}^n y_i^2},$$

where (λ_i, v_i) is the i-th eigenpair after orthonormalization and $y_i = v_i^* x$ is the ith coordinate of x in the eigenbasis. Let $\lambda_{\max} = \max \{\lambda_i\}_{i=1}^n$, due to the fact that the eigenvector is finite, we get

$$F(x) \leq \lambda_{\max} < \infty.$$

Combining with fact that $F(x) = F(x/|x|)$ for all $x \in \Omega$, we obtain

$$dom(F) = \{x/|x| \in \mathbb{R}^n \text{ s.t. } F(x/|x|) < \infty\} = \mathbb{R}^n.$$

As a consequence, $\underset{\Omega}{\operatorname{argmin}} F$ is not empty by $dom(F) = \mathbb{R}^n$. Similarly doing for $-F(x)$, we obtain $\underset{\Omega}{\operatorname{argmax}} F$ is not empty. □

(2) Compute $\nabla F(x)$ for $x \in \Omega$ and prove that $x^\top \nabla F(x) = 0$ for all $x \in \Omega$.

Proof. Calculate the gradient, and we get

$$\nabla F(x) = \frac{2Ax||x||^2 - x^\top Ax2x}{||x||^4} = \frac{2}{||x||^4} \left(Ax||x||^2 - x^\top Axx \right), \quad (1)$$

for $x \in \Omega$. Therefore

$$\begin{aligned} x^\top \nabla F(x) &= \frac{2}{||x||^4} x^\top \left(Ax||x||^2 - x(x^\top Ax) \right) = \frac{2}{||x||^4} \left(x^\top Ax||x||^2 - x^\top x(x^\top Ax) \right) \\ &= \frac{2}{||x||^4} \left(||x||^2 x^\top Ax - ||x||^2 x^\top Ax \right) = 0. \end{aligned}$$

□

(3) Prove that $\nabla F(x) = 0$ if and only if there exists $\lambda \in \mathbb{R}$ such that $Ax = \lambda x$.

Proof. Due to $\nabla F(x)$ is vector gradient, then if there exists $\lambda \in \mathbb{R}$ such that $Ax \neq \lambda x$, $\nabla F(x) \in \mathbb{R}^n$ so must be $\nabla F(x) \neq 0$. Otherwise, if $Ax = \lambda x$, replace in Equation 1, we have

$$\nabla F(x) = \frac{2}{||x||^4}\left(Ax||x||^2 - x^\top Axx\right) = \frac{2}{||x||^4}\left(\lambda x||x||^2 - (\lambda x)^\top xx\right) = \frac{2}{||x||^4}\left(\lambda x||x||^2 - \lambda||x||^2 x\right) = 0.$$

□

(4) Let

$$h(x) = Ax - \frac{x^\top Ax}{|x|^2}x$$

Prove that, when $\nabla F(x) \neq 0$, $-h(x)$ is a direction of descent for F at x.

Proof. From Equation 1, we have

$$\begin{aligned}
-h(x)^\top \nabla F(x) &= -\left(Ax - \frac{x^\top Ax}{||x||^2}x\right)^\top \left(\frac{2}{||x||^4}\left(Ax||x||^2 - x^\top Axx\right)\right) \\
&= \frac{2||x^\top Ax||^2}{||x||^4} - \frac{2||x^\top Ax||^2}{||x||^4} - \frac{2||Ax||^2}{||x||^2} + \frac{2||x^\top Ax||^2}{||x||^4} \\
&= \frac{2}{||x||^2}\left(-||Ax||^2||x||^2 + ||x^\top Ax||^2\right).
\end{aligned}$$

Apply Cauchy–Schwarz inequalities, we have $||x^\top Ax||^2 \leq ||Ax||^2||x||^2$, therefore

$$-h(x)^\top \nabla F(x) = \frac{2}{||x||^2}\left(-||Ax||^2||x||^2 + ||x^\top Ax||^2\right) \leq 0.$$

Combining with $\nabla F(x) \neq 0$, we obtain $-h(x)^\top \nabla F(x) < 0$. As a consequence, $-h(x)$ is a direction of descent for F at x. □

(5) Compute $\nabla^2 F(x)$ at $x \in \Omega$ and show that $x^\top \nabla^2 F(x) x = 0$ for all $x \in \Omega$.
From Equation 1, we have

$$\begin{aligned}
\nabla^2 F(x) &= \frac{2||x||^2 A - 4Axx^\top}{||x||^4} - \frac{6||x||^4 Axx^\top - 8x^\top Axxx^\top xx^\top}{||x||^8} \\
&= \frac{2||x||^6 A - 10||x||^4 Axx^\top + 8(x^\top Ax)xx^\top xx^\top}{||x||^8},
\end{aligned} \tag{2}$$

at $x \in \Omega$. Therefore for all $x \in \Omega$, we obtain

$$\begin{aligned}
x^\top \nabla^2 F(x) x &= \frac{2||x||^6 x^\top Ax - 10||x||^4 x^\top Axx^\top x + 8(x^\top Ax^\top)x^\top xx^\top xx^\top x}{||x||^8} \\
&= \frac{2||x||^6 x^\top Ax - 10||x||^6 x^\top Ax + 8||x||^6 x^\top Ax^\top}{||x||^8} = 0.
\end{aligned}$$

(6) Let $x \in \mathbb{R}^n$ be such that $Ax = \lambda x$ for some $\lambda \in \mathbb{R}$. Prove that $x \in \underset{\Omega}{\text{argminF}}$ requires that $A - \lambda Id_{\mathbb{R}^n} \succeq 0$ and $x \in \underset{\Omega}{\text{argmaxF}}$ that $A - \lambda Id_{\mathbb{R}^\ltimes} \preceq 0$.

Proof. Let $y \in \mathbb{R}^n$, from Equation 2, and due to $x \in \mathbb{R}^n$ be such that $Ax = \lambda x$ for some $\lambda \in \mathbb{R}$, we have

$$\begin{aligned}
y^\top \nabla^2 F(x) y &= \frac{2||x||^6 y^\top Ay - 10||x||^4 y^\top Axx^\top y + 8(x^\top Ax)y^\top xx^\top xx^\top y}{||x||^8} \\
&= \frac{2||x||^6 y^\top Ay - 10||x||^4 \lambda y^\top xx^\top y + 8\lambda x^\top xy^\top xx^\top xx^\top y}{||x||^8} \\
&= \frac{2||x||^6 y^\top Ay - 10||x||^6 \lambda y^\top y + 8||x||^6 \lambda y^\top y}{||x||^8} = \frac{2||x||^6 y^\top Ay - 2||x||^6 \lambda y^\top y}{||x||^8} \\
&= \frac{2}{||x||^2}\left[y^\top \left(A - \lambda Id_{\mathbb{R}^n}\right) y\right].
\end{aligned}$$

If $x \in \underset{\Omega}{\text{argminF}}$, then $y^\top \nabla^2 F(x) y \geq 0$, i.e.,

$$\frac{2}{||x||^2} \left[y^\top \left(A - \lambda Id_{\mathbb{R}^n} \right) y \right] \geq 0.$$

As a consequence, $A - \lambda Id_{\mathbb{R}^n} \succeq 0$. Similarly, if $x \in \underset{\Omega}{\text{argmaxF}}$, then $y^\top \nabla^2 F(x) y \leq 0$, so $A - \lambda Id_{\mathbb{R}^n} \preceq 0$. □

(7) For $x \in \Omega$, let

$$v(x) = \frac{|Ax|^2}{|x|^2} - F(x)^2.$$

Prove that $v(x) \geq 0$ for all x and that $v(x) = 0$ if and only if $\nabla F(x) = 0$.

Proof. We have

$$v(x) = \frac{||Ax||^2}{||x||^2} - \frac{||x^\top Ax||^2}{||x||^4} = \frac{||Ax||^2 ||x||^2 - ||x^\top Ax||^2}{||x||^4}.$$

Apply Cauchy–Schwarz inequalities, we have $||x^\top Ax||^2 \leq ||Ax||^2 ||x||^2$, therefore, we obtain

$$v(x) = \frac{||Ax||^2 ||x||^2 - ||x^\top Ax||^2}{||x||^4} \geq 0.$$

Let $\lambda \in \mathbb{R}$ such that $Ax = \lambda x$, we have $A - \lambda Id_{\mathbb{R}^n} \succeq 0$, so $F(x)^2$ is a strictly convex function. Similarly, $\frac{||Ax||^2}{||x||^2}$ is also convex, we obtain $v(x)$ is a strictly convex function and has a unique minimizer. Therefore, due to $\nabla F(x) = 0$ if and only if $Ax = \lambda x$, we obtain

$$v(x) = \frac{||\lambda x||^2 ||x||^2 - ||\lambda x^\top x||^2}{||x||^4} = \frac{\lambda^2 ||x||^4 - \lambda^2 ||x||^4}{||x||^4} = 0.$$

As a consequence, $v(x) = o$ if and only if $\nabla F(x) = 0$. □

(8) For $\alpha > 0$ and $x \in \Omega$, prove that $x - \alpha h(x) \in \Omega$.

Proof. Due to $x \in \Omega$, i.e., $x \in \mathbb{R}^n \setminus \{0\}$, we have

$$x - \alpha h(x) = x - \alpha \left(Ax - \frac{x^\top Ax}{||x||^2} x \right) = \frac{x||x||^2 - \alpha Ax ||x||^2 + \alpha (x^\top Ax) x}{||x||^2}. \tag{3}$$

Let $\lambda \in \mathbb{R}$ and consider 2 case where $Ax \neq \lambda x$ and $Ax = \lambda x$. From Equation 3, we have if $Ax \neq \lambda x$, then $x - \alpha h(x) \in \mathbb{R}^n$, otherwise, if $Ax = \lambda x$, since $\alpha > 0$, we have

$$x - \alpha h(x) = \frac{x||x||^2 - \alpha \lambda ||x||^2 x + \alpha \lambda ||x||^2 x}{||x||^2} = x,$$

therefore, we obtain $x - \alpha h(x) \in \mathbb{R}^n \setminus \{0\}$, i.e., $x - \alpha h(x) \in \Omega$. □

(9) For $\alpha > 0$ and $x \in \Omega$, let

$$x_\alpha = \frac{x - \alpha h(x)}{|x - \alpha h(x)|}.$$

Prove that, when $\nabla F(x) \neq 0$, $F(x_\alpha) < F(x)$ for small enough α, and that $\alpha \mapsto F(x_\alpha)$ is, when $|x| = 1$, minimized at

$$\alpha^*(x) = \frac{-(w(x) - F(x)v(x)) + \sqrt{(w(x) - F(x)v(x))^2 + 4v(x)^3}}{2v(x)^2}$$

with $w(x) = h(x)^\top Ah(x)$.

Proof. Due to when $\nabla F(x) \neq 0$, $-h(x)$ is a direction of descent for F at x. Following the definition of the direction of descent, for a small enough α, we have

$$F(x - \alpha h(x)) < F(x).$$

Since $x_\alpha = \frac{x-\alpha h(x)}{||x-\alpha h(x)||}$ and $|x - \alpha h(x)| > 0$, we obtain

$$F(x_\alpha) < F(x).$$

Let consider mapping $\alpha \mapsto F(x_\alpha)$, we have

$$\begin{aligned} F(x_\alpha) &= \frac{\left(\frac{x-\alpha h(x)}{||x-\alpha h(x)||}\right)^\top A \left(\frac{x-\alpha h(x)}{||x-\alpha h(x)||}\right)}{\left(\frac{x-\alpha h(x)}{||x-\alpha h(x)||}\right)^\top \left(\frac{x-\alpha h(x)}{||x-\alpha h(x)||}\right)} = \frac{(x-\alpha h(x))^\top A (x - \alpha h(x))}{(x-\alpha h(x))^\top (x-\alpha h(x))} \\ &= \frac{x^\top Ax - \alpha x^\top Ah(x) - \alpha h(x)^\top Ax + \alpha^2 h(x)^\top Ah(x)}{x^\top x - \alpha x^\top h(x) - \alpha h(x)^\top x + \alpha^2 h(x)^\top h(x)}. \end{aligned}$$

Due to $||x|| = 1$, take derivative, we obtain

$$\frac{d}{d\alpha}F(x_\alpha) = \frac{-2x^\top Ah(x) + 2\alpha h(x)^\top Ah(x) - 2\alpha^2 h(x)^\top Ah(x) x^\top h(x) + 2x^\top Axx^\top h(x) - 2\alpha x^\top Axh(x)^\top h(x)}{(x^\top x - 2\alpha x^\top h(x) + \alpha^2 h(x)^\top h(x))^2}.$$

Also since $||x|| = 1$, we obtain $F(x) = x^\top Ax$, $h(x) = Ax - (x^\top Ax)x = Ax - F(x)x$, and $v(x) = ||Ax||^2 - F(x)^2 = x^\top Ah(x)$. Combining with $w(x) = h(x)^\top Ah(x)$, we have

$$\begin{aligned} &\frac{d}{d\alpha}F(x_\alpha) = 0 \\ \Leftrightarrow\ & v(x)^2\alpha^2 + (w(x) - F(x)v(x))\alpha - v(x) = 0, \end{aligned} \tag{4}$$

so the discriminant is $\Delta = (w(x) - F(x)v(x))^2 + 4v(x)^3$. Therefore, one solution of Equation 4 is

$$\frac{-(w(x) - F(x)v(x)) + \sqrt{(w(x) - F(x)v(x))^2 + 4v(x)^3}}{2v(x)^2}.$$

As a consequence, the mapping $\alpha \mapsto F(x_\alpha)$ is minimized at

$$\alpha^*(x) = \frac{-(w(x) - F(x)v(x)) + \sqrt{(w(x) - F(x)v(x))^2 + 4v(x)^3}}{2v(x)^2}.$$

□

(10) Take $\epsilon = 10^{-6}$. Program an algorithm that takes as input a matrix A, an initial vector x_0 with $|x_0| = 1$ and a maximal number of iterations, N, and iterates

$$x_{t+1} = \frac{x_t - \alpha^*(x_t)h(x_t)}{|x_t - \alpha^*(x_t)h(x_t)|}$$

until $t = N$ or $|\nabla F(x)| < \epsilon$, whichever comes first.
Apply your algorithm to the matrix A in the file project2_A.csv, using $N = 2000$ and $x_0 = \mathbb{I}_n/\sqrt{n}$, where $\mathbb{I}_n$ is the vector with all coordinates equal to 1. Return the number of iterations, $t_{\max}$, needed by the algorithm and the final value of $F(x_t)$.
Plot the values of $F(x_t)$ as a function of t for $t = 0, \cdots, t_{\max}$.

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import torch

def func_F(x, A):
    nemonator = torch.matmul(torch.matmul(x.t(), A), x)
    denominator = torch.matmul(x.t(), x)
    out = nemonator/denominator
    return out

def grad_F(x, A):
    nemonator = 2 * torch.matmul(A, x) * torch.matmul(x.t(), x) - torch.matmul(torch.matmul(x.t(), A), x) * 2 * x
    denominator = torch.matmul(x.t(), x) ** 2
    out = nemonator/denominator
    return out

def func_h(x, A):
    nemonator = torch.matmul(torch.matmul(x.t(), A), x)
    denominator = torch.norm(x) ** 2
    out = torch.matmul(A, x) - (nemonator/denominator) * x
    return out

def func_w(x, A):
    h_x = func_h(x, A)
    return torch.matmul(torch.matmul(h_x.t(), A), h_x)

def func_v(x, A):
    term_1 = (torch.norm(torch.matmul(A, x)) ** 2) / (torch.norm(x) ** 2)
    term_2 = func_F(x, A) ** 2
    return term_1 - term_2

def func_alpha_star(x, A):
    w_x = func_w(x, A)
    v_x = func_v(x, A)
    F_x = func_F(x, A)
    nemonator = -(w_x - F_x * v_x) + torch.sqrt((w_x - F_x * v_x) ** 2 + 4 * (v_x ** 3))
    denominator = 2 * (v_x ** 2)
    return nemonator / denominator

if __name__ == "__main__":
    A = pd.read_csv('homework2_data/project2_A.csv')
    A = A.drop(['Unnamed: 0'], axis=1).to_numpy()
    A = torch.tensor(A)

    x = torch.ones(A.shape[0], dtype = torch.float64)
    x_0 = x/np.sqrt(A.shape[0])

    epsilon = 1e-6
    N = 2000
    t = 0
    list_t, list_f = [], []
    while True:
        if t == N or torch.norm(grad_F(x, A)) < epsilon:
            break
        list_f.append(func_F(x, A))
        list_t.append(t)
        tmp = x - func_alpha_star(x, A) * func_h(x, A)
        x = tmp/torch.norm(tmp)
        t += 1

    print("The number of required iterations: " + str(t))
    print("The value of the objective function at convergence: " + str(list_f[t-1].item()))

    plt.plot(list_t, list_f)
    plt.xlabel("t")
    plt.ylabel(r'$F(x_t)$')
    plt.title("Visualiztion of " + r'$F(x_t)$' + " as a function of t")
    plt.savefig("1.1.pdf")
```

Result:
The number of required iterations: 304
The value of the objective function at convergence: -13.574511277318216

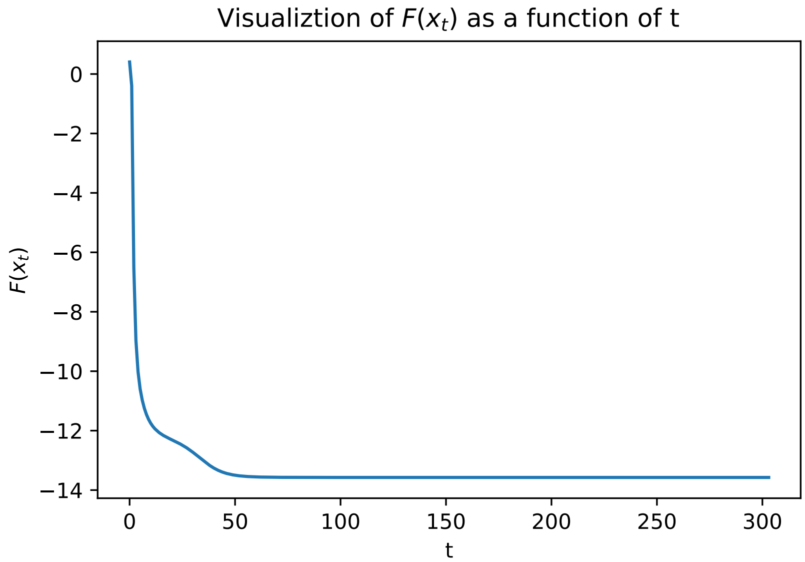

Figure 1: Visualization of $F(x_t)$ as a function of t for $t = 0, \cdots, t_{\max}$.

(11) Let $x^* \in \underset{\Omega}{\operatorname{argmin}} F$. Assume $x_0^\top x^* = 0$ and show that $x_t^T x^* = 0$ at each step of the preceding algorithm. Deduce from this that, under these assumptions, x_t cannot converge to x^*.
We have

$$\begin{aligned} x_{t+1}^\top x^* &= \frac{(x_t - \alpha^*(x_t)h(x_t))^\top}{||x_t - \alpha^*(x_t)h(x_t)||} x^* \\ &= \frac{x_t^\top x^* - \alpha^*(x_t)\left(x_t^\top A x^* - x_t^\top \left(\frac{x_t^\top A x_t}{||x_t||^2}\right) x^*\right)}{||x_t - \alpha^*(x_t)h(x_t)||}. \end{aligned}$$

Due to $x^* \in \underset{\Omega}{\operatorname{argmin}} F$, $\nabla F(x^*) = 0$ if and only if there exists $\lambda \in \mathbb{R}$ such that $Ax^* = \lambda x^*$, we obtain

$$x_{t+1}^\top x^* = \frac{x_t^\top x^* - \alpha^*(x_t)\left(x_t^\top \lambda x^* - x_t^\top \left(\frac{x_t^\top A x_t}{||x_t||^2}\right) x^*\right)}{||x_t - \alpha^*(x_t)h(x_t)||}. \tag{5}$$

For $t = 0$ and if $x_0^\top x^* = 0$, then Equation 5 shows $x_1^\top x^* = 0$, then if $x_1^\top x^* = 0$, $x_2^\top x^* = 0$. Continuously, we obtain $x_t^T x^* = 0$ at each step of the preceding algorithm.
Now, assume at step t, x_t converge to x^* and we will have

$$\begin{aligned} & x^* = x_t - \alpha^*(x_t)h(x_t) \\ \Leftrightarrow\ & x_t^\top x* = x_t^\top \left(x_t - \alpha^*(x_t)Ax_t - \alpha^*(x_t)\frac{x_t^\top A x_t}{||x_t||^2} x_t\right) \\ \Leftrightarrow\ & 0 = ||x_t||^2 - \alpha^*(x_t)x_t^\top A x_t + \alpha^*(x_t)\frac{x_t^\top A x_t}{||x_t||^2} x_t^\top x_t \\ \Leftrightarrow\ & ||x_t||^2 = \alpha^*(x_t)x_t^\top A x_t - \alpha^*(x_t)x_t^\top A x_t = 0 \text{ (contradiction with assumption } x \in \Omega). \end{aligned}$$

As a consequence, x_t cannot converge to x^*.

2 Problem 2

(1) Let $F : \mathbb{R}^n \mapsto \mathbb{R}$ be a C^1 function. Prove that if $x, u \in \mathbb{R}^n$ are such that $\nabla F(x)^\top u \neq 0$, then

$$h_u(x) = -(\nabla F(x)^\top u)u$$

is a direction of descent for F at x.

Proof. We have

$$\begin{aligned} h_u(x)^\top \nabla F(x) &= -\left((\nabla F(x)^\top u)u\right)^\top \nabla F(x) \\ &= -u^\top (\nabla F(x)^\top u) \nabla F(x) \\ &= -||\nabla F(x)^\top u||^2 < 0. \end{aligned}$$

Since $h_u(x)^\top \nabla F(x) < 0$, we obtain $-h(x)$ is a direction of descent for F at x. □

(2) Let $e_1, \cdots, e_n$ be the canonical basis of $\mathbb{R}^n$. Show that

$$h_{e_i}(x) = -\partial_{x_i} F(x) e_i.$$

Fix a small $\epsilon > 0$. Fix a sequence $(i_t, t \geq 0)$ with $i_t \in \{1, \cdots, N\}$. An algorithm that iterates

$$x_{t+1} = \begin{cases} x_t - \alpha_t \partial_{x_{i_t}} F(x_t) e_{i_t}, & \text{if } |\partial_{x_{i_j}} F(x_t)| \geq \epsilon \\ x_t, & \text{otherwise.} \end{cases}$$

is called a coordinate descent algorithm. This algorithm will be used in the next question.
We have

$$\begin{aligned} h_{e_i}(x) &= -\left(\nabla F(x)^\top e_i\right) e_i \\ &= -\left((\partial_{x_1} F(x), \cdots, \partial_{x_n} F(x))^\top e_i\right) e_i. \end{aligned}$$

Let $e_i = (e_{i_1}, \cdots, e_{i_n})$, due to $e_1, \cdots, e_n$ are the canonical basis of $\mathbb{R}^n$, for $j \in \{1, \cdots, n\}$, we have

$$\partial_{x_j} F(x) e_{i_j} = \begin{cases} \partial_{x_i} F(x), & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we obtain

$$\begin{aligned} h_{e_i}(x) &= -\left((\partial_{x_1} F(x), \cdots, \partial_{x_n} F(x))^\top e_i\right) e_i \\ &= -\left(\sum_{j=0}^{n} \partial_{x_j} F(x) e_{i_j}\right) e_i = -\partial_{x_i} F(x) e_i. \end{aligned}$$

3 Problem 3

(1) Let $I \in \mathbb{R}$ be an interval. Prove that, if $f : I \mapsto \mathbb{R}$ is convex and non-decreasing, and $\varphi : \mathbb{R}^n \mapsto I$ is convex, then $F = f \circ \varphi$ is convex.

Proof. Due to $\varphi : \mathbb{R}^n \mapsto I$ is convex on $\mathbb{R}^n$, we have

$$\varphi(\lambda x_1 + (1-\lambda) x_2) \leq \lambda \varphi(x_1) + (1-\lambda)\varphi(x_2),$$

$\forall x_1, x_2 \in \mathbb{R}^n$, and $\lambda \in [0, 1]$. Moreover, since $f : I \mapsto \mathbb{R}$ is non-decreasing, we get

$$f(\varphi(\lambda x_1 + (1-\lambda) x_2)) \leq f(\lambda \varphi(x_1) + (1-\lambda)\varphi(x_2)). \tag{6}$$

Additionally, due to $f : I \mapsto \mathbb{R}$ is also convex on the interval $I \in \mathbb{R}$, we get

$$f(\lambda \varphi(x_1) + (1-\lambda)\varphi(x_2)) \leq \lambda f(\varphi(x_1)) + (1-\lambda) f(\varphi(x_2)). \tag{7}$$

Combining the result from Inequality 6 and 7, we obtain

$$f(\varphi(\lambda x_1 + (1-\lambda) x_2)) \leq \lambda f(\varphi(x_1)) + (1-\lambda) f(\varphi(x_2)),$$

$\forall x_1, x_2 \in \mathbb{R}^n$, and $\lambda \in [0, 1]$. As a consequence, $F = f \circ \varphi$ is convex on $\mathbb{R}^n$. □

(2) Prove that $\Psi : u \mapsto \log\cosh(|u|)$ is C^1 and convex on $\mathbb{R}^n$ and give the expression of $\nabla\Psi(u)$.

Proof. We have

$$\Psi(u) = \log\cosh(||u||) = \log\frac{e^{||u||}+e^{-||u||}}{2}.$$

Calculate the gradient, and we get

$$\nabla\Psi(u) = \frac{e^{||u||}-e^{-||u||}}{e^{||u||}+e^{-||u||}}\mathbb{I}_n,$$

where $\mathbb{I}_n$ is the vector with all coordinates equal to 1. Due to the denominator $e^{||u||}+e^{-||u||} > 0$, $\forall u \in \mathbb{R}^n$, then $\Psi(u)$ is differentiable on $\mathbb{R}^n$ and its gradient $\nabla\Psi(u)$ is continuous on $\mathbb{R}^n$. Therefore, we obtain $\Psi : u \mapsto \log\cos h(||u||)$ is C^1.
Let consider $u_1, u_2 \in \mathbb{R}^n$ and $\lambda \in [0,1]$, we have

$$\Psi(\lambda u_1 + (1-\lambda)u_2) = \log\left(\frac{e^{||\lambda u_1+(1-\lambda)u_2||}+e^{-||\lambda u_1+(1-\lambda)u_2||}}{2}\right),$$

and

$$\begin{aligned}\lambda\Psi(u_1)+(1-\lambda)\Psi(u_2) &= \lambda\left(\log\frac{e^{||u_1||}+e^{-||u_1||}}{2}\right)+(1-\lambda)\left(\log\frac{e^{||u_2||}+e^{-||u_2||}}{2}\right)\\ &= \log\left(\frac{\left(e^{||u_1||}+e^{-||u_1||}\right)^{\lambda}\left(e^{||u_2||}+e^{-||u_2||}\right)^{1-\lambda}}{2}\right).\end{aligned}$$

Since $\lambda \in [0,1]$, apply the Binomial theorem for $(a+b)^\lambda$, $\forall a, b \in \mathbb{R}$, and we get

$$e^{||\lambda u_1+(1-\lambda)u_2||}+e^{-||\lambda u_1+(1-\lambda)u_2||} \le \left(e^{||u_1||}+e^{-||u_1||}\right)^{\lambda}\left(e^{||u_2||}+e^{-||u_2||}\right)^{1-\lambda}.$$

Combining with the fact that to $\log(x)$ is a convex and monotonically non-decreasing function, we obtain

$$\log\left(\frac{e^{||\lambda u_1+(1-\lambda)u_2||}+e^{-||\lambda u_1+(1-\lambda)u_2||}}{2}\right) \le \log\left(\frac{\left(e^{||u_1||}+e^{-||u_1||}\right)^{\lambda}\left(e^{||u_2||}+e^{-||u_2||}\right)^{1-\lambda}}{2}\right), \tag{8}$$

i.e., $\Psi(\lambda u_1 + (1-\lambda)u_2) \le \lambda\Psi(u_1)+(1-\lambda)\Psi(u_2)$, $\forall u_1, u_2 \in \mathbb{R}^n$, and $\lambda \in [0,1]$. As a consequence, $\Psi : u \mapsto \log\cosh(||u||)$ is convex on $\mathbb{R}^n$. □

(3) Assume that an integer d, and a set $\mathcal{L}$ of non-ordered pairs $\{i,j\}$, with $1 \le i \ne j \le d$ are given. Let Ω be the vector space of all vectors indexed by $\mathcal{L}$, i.e., the set of all

$$x = (x_{\{i,j\}}, \{i,j\} \in \mathcal{L}).$$

Alternatively, $x \in \Omega$ can be seen as a $d \times d$ symmetric matrix such that $x_{ij} = 0$ if $\{i,j\} \notin \mathcal{L}$. To lighten the notation, we write below $x_\ell = x_{ij}$ for $\ell = \{i,j\} \in \mathcal{L}$.
Assume that a training set of vectors $y_1, \cdots, y_N \in \mathbb{R}^d$ is observed. Define, for $x \in \Omega$, considered as a $d \times d$ matrix,

$$F(x) = \sum_{k=1}^{N}\Psi(y_k - xy_k).$$

Prove that F is a convex function of x.

Proof. Let consider $x_1, x_2 \in \Omega$ and $\lambda \in [0,1]$, we have

$$F(\lambda x_1 + (1-\lambda)x_2) = \sum_{i=1}^{N}\Psi\left(y_k - (\lambda x_1 + (1-\lambda)x_2)\, y_k\right) = \sum_{i=1}^{N}\Psi\left(y_k - \lambda x_1 y_k - x_2 y_k + \lambda x_2 y_k\right).$$

On the other hand, we also have

$$\lambda F(x_1) + (1-\lambda)F(x_2) = \sum_{i=1}^{N} \lambda\Psi(y_k - x_1 y_k) + (1-\lambda)\Psi(y_k - x_2 y_k)$$

$$= \sum_{i=1}^{N} \log\left(\frac{\left(e^{||y_k - x_1 y_k||} + e^{-||y_k - x_1 y_k||}\right)^{\lambda}\left(e^{||y_k - x_2 y_k||} + e^{-||y_k - x_2 y_k||}\right)^{1-\lambda}}{2}\right)$$

$$\geq \underbrace{\sum_{i=1}^{N} \log\left(\frac{e^{||\lambda(y_k - x_1 y_k) + (1-\lambda)(y_k - x_2 y_k)||} + e^{-||\lambda(y_k - x_1 y_k)+(1-\lambda)(y_k - x_2 y_k)||}}{2}\right)}_{\Psi(\lambda(y_k - x_1 y_k) + (1-\lambda)(y_k - x_2 y_k)) = \Psi(y_k - \lambda x_1 y_k - x_2 y_k + \lambda x_2 y_k)} \text{(since Inequality 8 and } e^x > 0, \forall x \in \mathbb{R}).$$

Therefore, we obtain

$$\sum_{i=1}^{N} \Psi\left(y_k - \lambda x_1 y_k - x_2 y_k + \lambda x_2 y_k\right) \leq \sum_{i=1}^{N} \lambda\Psi(y_k - x_1 y_k) + (1-\lambda)\Psi(y_k - x_2 y_k),$$

i.e.,

$$F(\lambda x_1 + (1-\lambda)x_2) \leq \lambda F(x_1) + (1-\lambda)F(x_2),$$

$\forall x_1, x_2 \in \Omega$, and $\lambda \in [0,1]$. As a consequence, F is convex of x on Ω. □

(4) Prove that

$$\partial_{x_{ij}} F(x) = -\sum_{k=1}^{N} \frac{\tanh(|z_k|)}{|z_k|}\left(z_k^{(i)} y_k^{(j)} + z_k^{(j)} y_k^{(i)}\right)$$

with $z_k = y_k - x y_k$.

Proof. Take partial derivative over x_{ij}, we obtain

$$\begin{aligned}
\partial_{x_{ij}} F(x) &= \partial_{x_{ij}} \sum_{k=1}^{N} \Psi(y_k - x y_k) = \sum_{k=1}^{N} \partial_{x_{ij}} \log\left(\cosh(||y_k - x y_k||)\right) \\
&= -\sum_{k=1}^{N} \frac{\sinh ||y_k - x y_k||}{\cosh(||y_k - x y_k||)} \partial_{x_{ij}} ||y_k - x y_k|| = -\sum_{k=1}^{N} \tanh ||y_k - x y_k|| \frac{\partial_{x_{ij}}(y_k - x y_k)}{||y_k - x y_k||} \\
&= -\sum_{k=1}^{N} \frac{\tanh ||y_k - x y_k||}{||y_k - x y_k||}\left(\left(y_k^{(i)} - x^{(i)} y_k^{(i)}\right) y_k^{(j)} + \left(y_k^{(j)} - x^{(j)} y_k^{(j)}\right) y_k^{(i)}\right) \\
&= -\sum_{k=1}^{N} \frac{\tanh(||z_k||)}{||z_k||}\left(z_k^{(i)} y_k^{(j)} + z_k^{(j)} y_k^{(i)}\right).
\end{aligned}$$

□

(5) Write a program that reads the vectors $y_1, \cdots, y_N$ in the file project2_Y.csv (with $N = 50$ and $n = 100$) and the locations of the non-zero entries of x in project_C.csv and runs a gradient descent algorithm to minimize F.
Your program should define the sequence $x(t) \in \Omega$ satisfying

$$x(t+1) = x(t) - \alpha_t \nabla F(x(t))$$

initialized with $x(0) = 0$ (the zero matrix). The coefficient α_t must be obtained using a backtracking line search, letting $\alpha_t = \bar{\alpha}\rho^{r_t}$ where r_t is the the smallest integer such that

$$F(x(t) - \alpha_t \nabla F(x(t))) \leq F(x(t)) - c_1 \alpha_t |\nabla F(x(t))|^2.$$

You will take $c_1 = 0.01$, $\bar{\alpha} = 0.1$ and $\rho = 0.9$.
You will stop the program as soon as $F(x(t-1)) - F(x(t)) \leq 10^{-6}$.

To describe the output of your program, provide the value of F at convergence, the largest element of x in absolute value, and the number of iterations required.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import torch

def func_Psi(u):
    return torch.log(torch.cosh(torch.norm(u)))

def func_F(matrix_x, list_y):
    out = 0
    for y_k in list_y:
        out += func_Psi(y_k - torch.matmul(matrix_x, y_k))
    return out

def grad_F(matrix_x, list_y):
    matrix_x_clone = matrix_x.clone()
    matrix_x_clone = matrix_x_clone.detach().requires_grad_()
    out = func_F(matrix_x_clone, list_y)
    out.backward()
    return matrix_x_clone.grad

def line_search(alpha_bar, rho, c1, matrix_x, list_y):
    r_t = 0
    while True:
        alpha = alpha_bar * (rho ** r_t)
        grad =  grad_F(matrix_x, list_y)
        lhs = func_F(matrix_x - alpha * grad, list_y)
        rhs = func_F(matrix_x, list_y) - c1 * alpha * (torch.norm(grad) ** 2)
        if lhs <= rhs:
            break
        else:
            r_t += 1
    return alpha

def find_max_abs_of_X(matrix_x):
    out = torch.abs(matrix_x[0][0])
    for rows in matrix_x:
        for entry in rows:
            if torch.abs(entry) > out:
                out = torch.abs(entry)
    return out

def problem_3_5(list_y):
    matrix_x = torch.zeros(list_y.shape[1], list_y.shape[1], dtype = torch.float64)
    t, c1, alpha_bar, rho = 0, 0.01, 0.1, 0.9
    while True:
        alpha = line_search(alpha_bar, rho, c1, matrix_x, list_y)
        matrix_x_new = matrix_x - alpha * grad_F(matrix_x, list_y)
        if func_F(matrix_x, list_y) - func_F(matrix_x_new, list_y) <= 1e-6:
            break
        else:
            matrix_x = matrix_x_new
            t += 1
    print("The value of F at convergence: " + str(func_F(matrix_x_new, list_y).item()))
    print("The largest element of x in absolute value: "+ str(find_max_abs_of_X(matrix_x_new).item()))
    print("The number of required iterations: " + str(t))

if __name__ == "__main__":
    list_y = pd.read_csv('homework2_data/project2_Y.csv')
    list_y = list_y.drop(['Unnamed: 0'], axis=1).to_numpy()
    list_y = torch.tensor(list_y, dtype = torch.float64)

    problem_3_5(list_y)
```

Result:
The value of F at convergence: 0.0001609264200027296
The largest element of x in absolute value: 0.7562430492024295
The number of required iterations: 344

(6) Order the elements of $\mathcal{L}$ as $\ell_1, \cdots, \ell_m$ as they are listed in the file project2_C.csv. Let $q_t = \ell_{j+1}$ if $t = j(\mod m)$, i.e., j is the remainder of the division of t by m, so that q_t explores periodically all the pairs in $\mathcal{L}$. For $\ell = \{i, j\} \in \mathcal{L}$, let $\xi^{(\ell)} \in \Omega$ be defined by $\xi^{(\ell)}_{ij} = 1$ and $\xi^{(\ell)}_{i'j'} = 0$ is $\{i', j'\} \neq \ell$.
Program the coordinate descent algorithm in Question 2, taking $\epsilon = 10^{-8}$ and

$$x(t+1) = \begin{cases} x(t) - \alpha_t \partial_{x_{q_t}} F(x(t)) \xi^{(q_t)}, & \text{if } |\partial_{x_{q_t}} F(x(t))| \geq \epsilon \\ x(t), & \text{otherwise.} \end{cases}$$

You will determine α_t using the same method as in Question 3.5, taking $\bar{\alpha} = 1$. You will stop the algorithm as soon as $F(x(t-m)) - F(x(t)) \leq 10^{-6}$ (therefore using the difference between two full sweeps of coordinates).
Run your program using the data in project2_Y.csv. To describe its output, provide the value of F at convergence (hopefully the same as in Question 3.5), the largest element of x in absolute value, and the number of iterations required.

```
def get_q_t(list_l, t, m):
    j = t % m
    q_t = list_l[j+1]
    return q_t

def get_xi_l(l, dims):
    x_i_l = torch.zeros(dims, dims, dtype = torch.float64)
    x_i_l[l[0], l[1]] = 1
    x_i_l[l[1], l[0]] = 1
    return x_i_l

def partial_F(q_t, matrix_x, list_y):
    out = 0
    i, j = q_t[0], q_t[1]
    for y_k in list_y:
        z_k = y_k - torch.matmul(matrix_x, y_k)
        out += (torch.tanh(torch.norm(z_k))/torch.norm(z_k)) * (z_k[i] * y_k[j] + z_k[j] * y_k[i])
    return -out

def problem_3_6(list_y, list_l):
    matrix_x = torch.zeros(list_y.shape[1], list_y.shape[1], dtype = torch.float64)
    t, c1, alpha_bar, rho = 0, 0.01, 1, 0.9
    list_x = []
    m = len(list_l) - 1
    while True:
        q_t = get_q_t(list_l, t, m)
        alpha = line_search(alpha_bar, rho, c1, matrix_x, list_y)
        partial = partial_F(q_t, matrix_x, list_y)
        if torch.abs(partial) >= 1e-8:
            matrix_x = matrix_x - alpha * partial * get_xi_l(q_t, list_y.shape[1])
        list_x.append(matrix_x)
        if t % 100 == 0:
          print("The value of F at step " + str(t) + ": " + str(func_F(matrix_x, list_y).item()))
        if t >= m and func_F(list_x[t - m], list_y) - func_F(list_x[t], list_y) <= 1e-6:
            break
        else:
            t += 1
    print("The value of F at convergence: " + str(func_F(matrix_x, list_y).item()))
    print("The largest element of x in absolute value: "+ str(find_max_abs_of_X(matrix_x).item()))
    print("The number of required iterations: " + str(t))

if __name__ == "__main__":
    list_l = pd.read_csv('homework2_data/project2_C.csv')
    list_l = list_l.drop(['Unnamed: 0'], axis=1).to_numpy()
    list_l = torch.tensor(list_l)

    problem_3_6(list_y, list_l)
```

Result:
The value of F at convergence: 14.627268743106319
The largest element of x in absolute value: 0.7640178922678903
The number of required iterations: 64133