Homework Assignment #1

Zixuan Ni

September 2023

1 Problem 1

(a) Machine learning is we use methods and algorithms to focus on processing data by decreasing human interaction.
(b) AI is the technology that is used to build machines and computers to have the ability to generate, cognitive things with human intelligence. However, Machine learning is just a subset of AI that automatically use a machine to learn and improve a single task. Machine learning is only for training and then analyzing but AI can do more than that.

2 Problem 2

(a) Code:

```
from numpy import array
from numpy.linalg import norm
import numpy as np
temp = array([5.4, -1.2, 0.0, 3.2])
l1 = norm(temp, 1)
l2 = norm(temp, 2)
l3 = norm(temp, 3)
l_inf = norm(temp, np.inf)
print(f"We have the 1-norm as {l1}, 2-norm as {l2}, 3-norm as {l3}, then we have infinity-norm as {l_inf}")

We have the 1-norm as 9.8, 1-norm as 6.390618123468183, 3-norm as 5.768597628524599, then we have infinity-norm as 5.4
```

(b) Code:

```
y = array([-1, 2.2, 3.0, 0.5])
l1 = norm(x - y, 1)
l2 = norm(x - y, 2)
l_inf = norm(x - y, np.inf)
print(f"We have the 1-norm as {l1}, 2-norm as {l2}, then we have infinity-norm as {l_inf}")

We have the 1-norm as 15.5, 2-norm as 8.295179322956196, then we have infinity-norm as 6.4
```

3 Problem 3

(a) We have A as $\begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$, we have the identity matrix as $\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$.
According to the rule that $A - \lambda I = 0$, we have $\begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$ - $\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$ which is $\begin{bmatrix} 5-\lambda & 4 \\ 4 & 5-\lambda \end{bmatrix}$.
To solve the matrix, we get the equation

$$(5-\lambda)^2 - 16 = 0$$
$$(5-\lambda)^2 = 16$$
$$(5-\lambda) = \pm 4$$

We will have λ as 9 and 1. For eigenvectors, we need to recall $\begin{bmatrix} 5-\lambda & 4 \\ 4 & 5-\lambda \end{bmatrix}$ and use values of λ.
When λ equals to 9, the matrix is $\begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix}$. We need to solve that

$$\begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The corresponding eigenvector is in the following relationship: $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ so an eigenvector is $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

When λ equals to 1, the matrix is $\begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$. We need to solve that

$$\begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The corresponding eigenvector is in the following relationship: $\begin{bmatrix} x_1 \\ -x_2 \end{bmatrix}$ so an eigenvector is $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

(b) Since we know the value of λ and two eigenvectors we can come up with the eigendecomposition in following:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

(c) Code:

```
B = np.array([[-13, -8, -4], [12, 7, 4], [24, 16, 7]])
w, v = np.linalg.eig(B)
print(f"The eigenvalues of B is {w} and the correspongding eigenvectors of B are {v}")

The eigenvalues of B is [-1.  3. -1.] and the correspongding eigenvectors of B are [[-0.51214752 -0.40824829 -0.02464807]
 [ 0.38411064  0.40824829 -0.41725537]
 [ 0.76822128  0.81649658  0.90845497]]
```

(d) A is positive-definite because both eigenvalues are positive. B is not because there are two negative eigenvalues.

4 Problem 4

Show convexity of $f(x) = a^T x + b$

$$\begin{aligned} & f(\lambda x_1 + (1-\lambda)x_2) \\ &= a^T(\lambda x_1 + (1-\lambda)x_2) + b \\ &= a^T \lambda x_1 + a^T x_2 - a^T \lambda x_2 + b \\ &\leq \lambda f(x_1) + (1-\lambda) f(x_2) \end{aligned}$$

5 Problem 5

(a) First we have the derivative respect to x and y as follows

$$2x - y$$

$$2y - x$$

Then we have the Hessian Matrix as follows

$$\begin{bmatrix} 2 & something \\ something & 2 \end{bmatrix}$$

Then we calculate the second partial derivative:

$$f_{xy}(x, y), f_{yx}(x, y)$$

So Hessian Matrix will be

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

```
H = np.array([[2, -1], [-1, 2]])
w, v = np.linalg.eig(H)
print(f"The eigenvalues of B is {w} and the correspongding eigenvectors of B are {v}")

The eigenvalues of B is [3. 1.] and the correspongding eigenvectors of B are [[ 0.70710678  0.70710678]
 [-0.70710678  0.70710678]]
```

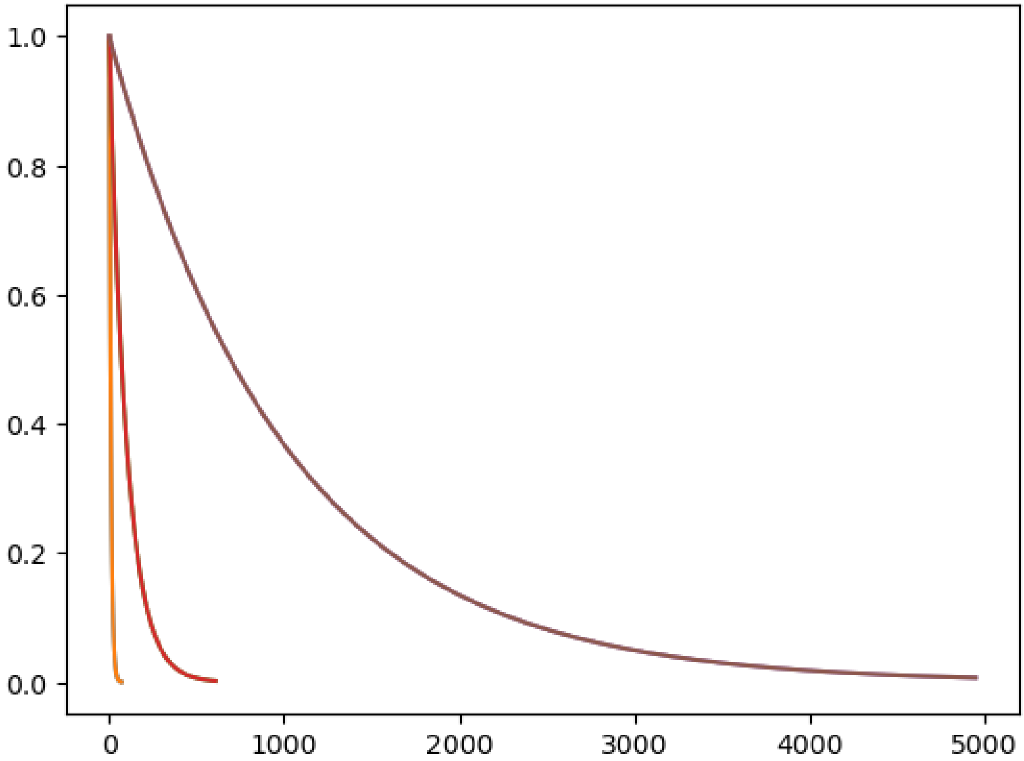
Since all λ values are positive, it is positive definite. (b) Code:

```python
import matplotlib.pyplot as plt
#methods and logic of the algo is from the course material "Python Tutorial_FA2023.pdf" which presented
#on class and uploaded on Canvas for reviewing. I have not and will not copy from material or classmates' work.
#I will cite every source.
#I guarantee the code is my own work and I totally understand the code, and it will not be copied by anyone.
#Based on my understanding of the code and lecture material, I can reproduce the code and algo without any help.
#I agree to the terms in Honor Code.
def func(x) :
    #get the update value of the curve/function after each iteration
    return x[0] ** 2 + x[1] ** 2 - x[0] * x[1]

def grad(x):
    #get the update value of deriv after each iteration and then return as array(matrix) form for next
    #iteration
    deriv0 = 2 * x[0] - x[1]
    deriv1 = 2 * x[1] - x[0]
    return np.array([deriv0, deriv1])

def main(grad, current_x, rate, precision, threshold, records):
        #use for loop to do iterations
        for i in range(threshold):
             #get the gradient value
             current_grad = grad(current_x)
             records.append(current_x)
             temp_x = current_x
            #use learning rate to get the current value
             current_x = current_x - current_grad * rate
             #follows the rule we learned on class: w_k = w_k-1 - ng'(w_k-1)
             if abs(func(temp_x) - func(current_x)) < precision:
                  break
        value = func(current_x)
        print("local minimum is "f'{value:.20f}')
        plt.plot(records)
        return records
main(grad, np.array([1, 1]), 0.1, 0.0000001, 10000, [])
main(grad, np.array([1, 1]), 0.01, 0.0000001, 10000, [])
main(grad, np.array([1, 1]), 0.001, 0.0000001, 10000, [])

local minimum is 0.00000039260092771818
local minimum is 0.00000492489108720544
local minimum is 0.00004982696258168871
#detailed graphs are in the .ipynb in the zip
```



1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000