1 a)

$f : \mathbb{R}^2 \mapsto \mathbb{R}$, $f(x_1,x_2) = x_1^4 + 2x_2^4 - 4x_1x_2$

$$\left\{ \begin{array}{lll} f'_{x_1} = 4x_1^3 - 4x_2 & f'_{x_2} = 8x_2^3 - 4x_1 & \\ f''_{x_1x_1} = 12x_1^2 & f''_{x_2x_2} = 24x_2^2 & f''_{x_1x_2} = -4 \end{array} \right.$$

$$\nabla f(\vec{x}) = \begin{bmatrix} 4x_1^3 - 4x_2 \\ 8x_2^3 - 4x_1 \end{bmatrix}$$

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} 12x_1^2 & -4 \\ -4 & 24x_2^2 \end{bmatrix}$$

$$\nabla f(\vec{x}) = 0 \Rightarrow \begin{cases} 4x_1^3 - 4x_2 = 0 \\ 8x_2^3 - 4x_1 = 0 \end{cases} \Rightarrow \begin{cases} x_1^3 - x_2 = 0 \\ -x_1 + 2x_2^3 = 0 \end{cases}$$

$$\begin{cases} x_1^3 - x_2 = 0 \\ -x_1 + 2x_2^3 = 0 \end{cases} \rightarrow \begin{cases} x_2 = x_1^3 & \textcircled{1} \\ -x_1 + 2x_2^3 = 0 & \textcircled{2} \end{cases}$$

$$\textcircled{2} \; -x_1 + 2(x_1^3)^3 = 0 \rightarrow -x_1 + 2x_1^9 = 0$$

$$\rightarrow x_1(2x_1^8 - 1) = 0 \quad \Leftarrow \boxed{x_1 = 0} \Rightarrow \boxed{x_2 = 0}$$

$$2x_1^8 - 1 = 0 \rightarrow x_1^8 = \frac{1}{2} \rightarrow x_1 = \pm\frac{1}{2^{1/8}}$$

$$\boxed{x_1 = \frac{1}{2^{1/8}}} : \Rightarrow x_2 = \frac{1}{2^{3/8}}$$

$$\boxed{x_1 = -\frac{1}{2^{1/8}}} \Rightarrow x_2 = -\frac{1}{2^{3/8}}$$

- Gradient is zero when we have the following vectors:

$$\vec{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \vec{x}_2 = \begin{bmatrix} 2^{-1/8} \\ 2^{-3/8} \end{bmatrix} \quad \vec{x}_3 = \begin{bmatrix} -2^{-1/8} \\ -2^{-3/8} \end{bmatrix}$$

1 matrix $M \in \mathbb{R}^{n\times n}$ is positive definite $\Leftrightarrow x^T M x > 0$ $\forall x \in \mathbb{R}^n \setminus \{0\}$. This holds if all eigenvalues of M are greater than zero. the matrix is ~~no~~ negative definite if $x^T M x < 0$, ~~the~~ all eigenvals must be less then zero.

- Sufficient conditions for local optimum
  If $f: \mathbb{R}^n \mapsto \mathbb{R}$ is a second order continous differentiable function:

  1. $\nabla f(x^*) = 0$
  2. $\nabla^2 f(x^*) > 0$ $\Rightarrow x^*$ is local minimizer

---

$$\nabla^2 f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & -4 \\ -4 & 0 \end{bmatrix} \rightarrow$$ This matrix is not PD, hence it's not a local minimizer

---

$$\nabla^2 f\left(\begin{bmatrix} 2^{-1/8} \\ 2^{-3/8} \end{bmatrix}\right) = \begin{bmatrix} \frac{12}{2^{1/4}} & -4 \\ -4 & \frac{24}{2^{6/8}} \end{bmatrix} \rightarrow \begin{matrix} \lambda_1 \approx 16{,}7 > 0 \\ \lambda_2 \approx 7{,}7 > 0 \end{matrix}$$

Since both eigenvalues are greater than zero, and the gradient in $x = [2^{-1/8}, 2^{-3/8}]$ is zero, we know that the point is a local minimizer

---

~~$\nabla^2 f\left(\begin{bmatrix} -2^{-1/8} \\ -2^{-3/8} \end{bmatrix}\right) = \nabla^2 f\left(\begin{bmatrix} 2^{-1/8} \\ 2^{-3/8} \end{bmatrix}\right) \Rightarrow$ The hessian is the s~~

$$\nabla^2 f\left(\begin{bmatrix} -2^{-1/8} \\ -2^{-3/8} \end{bmatrix}\right) = \nabla^2 f\left(\begin{bmatrix} 2^{-1/8} \\ 2^{-3/8} \end{bmatrix}\right)$$

$\Rightarrow$ the hessian is the same for

$\vec{x}_2 = \begin{bmatrix} 2^{-1/8} \\ 2^{-3/8} \end{bmatrix}$ and $\vec{x}_3 = \begin{bmatrix} -2^{-1/8} \\ -2^{-3/8} \end{bmatrix}$

This means that $\nabla^2 f(\vec{x}_3)$ also has eigenvalues that are greater than zero, which means that it is also a local minimizer.

$\vec{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ is not a local maximum or minimum since it has one positive and one negative eigenvalue, The hessian in the point $\vec{x}_1$ can therefore be neither positive definite nor negative definite

Answer: $\vec{x}_2 = \begin{bmatrix} 2^{-\frac{1}{8}} \\ 2^{-3/8} \end{bmatrix}$ and $\vec{x}_3 = \begin{bmatrix} -2^{-1/8} \\ -2^{-3/8} \end{bmatrix}$ are local minimizers

$\vec{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ is neither a local minimizer nor maximizer

1b) $f : \mathbb{R}^2 \mapsto \mathbb{R}$, $f(\vec{x}) = \vec{x}^T \cdot A \vec{x} + b^T \vec{x}$

$$A = \begin{bmatrix} -1 & 0 & 1/2 \\ 0 & -1 & 0 \\ 1/2 & 0 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

$$\frac{\partial f(\vec{x})}{\partial \vec{x}} = (A + A^T)\vec{x} + b\vec{x}$$

$$\frac{\partial f(\vec{x})}{\partial \vec{x}} = \begin{bmatrix} -2 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & -2 \end{bmatrix} \vec{x} + \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} = 0$$

$$\begin{cases} \textcircled{1} & -2x_1 + x_3 = 0 \\ \rightarrow \textcircled{2} & -2x_2 = -1 \rightarrow \boxed{x_2 = \frac{1}{2}} \\ \textcircled{3} & x_1 - 2x_3 = -2 \end{cases}$$

$$\textcircled{1} + 2 \cdot \textcircled{3} \Rightarrow -3x_3 = -4 \Rightarrow \boxed{x_3 = \frac{4}{3}}$$

$$x_1 = -2 + 2 \cdot \frac{4}{3} \rightarrow \boxed{x_1 = \frac{2}{3}}$$

- Gradient $\nabla f(\vec{x}) = 0$ when $\vec{x} = \begin{bmatrix} 2/3 \\ 1/2 \\ 4/3 \end{bmatrix}$

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} -2 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & -2 \end{bmatrix}$$

eigenvalues of $\nabla^2 f(\vec{x})$ are:
$\lambda_1 = -1$, $\lambda_2 = -2$, $\lambda_3 = -3$

- The hessian is Negative definite in the point $\vec{x} = [2/3 \;\; 1/2 \;\; 4/3]^T$ since all it's eigenvalues are less then zero, in the point.

- Because the gradient is also $\nabla f(\vec{x}) = 0$ in the point we know that it's a local maximizer.

Answer: $\vec{x} = [2/3 \;\; 1/2 \;\; 4/3]^T$ is a local maximizer

2. $f : \mathbb{R}^n \mapsto \mathbb{R}$, $f(\vec{x}) = \|Ax - b\|_2^2$,

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$

$$\begin{cases} x^{(0)} = a \; , \; a \in \mathbb{R}^n & \text{for } t=0 \\ x^{(t+1)} = x^{(t)} - \eta \cdot \nabla f(x^{(t)}), & \text{for } t>0 \end{cases}$$

$$\frac{\partial f}{\partial x} = 2(Ax - B)^t A$$

$$\begin{cases} x^{(t)} = a \quad \text{where } a \in \mathbb{R}^n & \text{for } t=0 \\ x^{(t+1)} = x^{(t)} - \eta \cdot 2(Ax^{(t)} - B)^t A & \text{for } t>0 \end{cases}$$

3. $$F(\omega) = \frac{1}{N}\sum_{i=1}^{N}\log\left(1+e^{-\langle\omega,x_i\rangle y_i}\right)$$

$$\nabla F(\omega) = \frac{1}{N}\sum_{i=1}^{N}\frac{-1}{1+e^{-\langle\omega,x_i\rangle y_i}}\cdot e^{-y_i\langle\omega,x_i\rangle}\cdot y_i x_i$$

$$= -\frac{1}{N}\sum\frac{e^{-y_i\sum_{a=1}^{k}w_a x_{ia}}\,y_i x_i}{1+e^{-y_i\sum_a w_a x_{ia}}}\quad,\quad b_i = -y_i\sum_{a=1}^{k}w_a x_{ia}\quad,$$

$$\nabla_b F(b) = \begin{bmatrix}\frac{d}{db}\sum\left(\frac{y_i x_{i1}e^{b_i}}{1+e^{a_i}}\right)\\ \vdots \\ \frac{d}{db}\left(\sum\frac{y_i x_{in}e^{a_i}}{1+e^{b_i}}\right)\end{bmatrix}(-N^{-1}) = \begin{bmatrix}\sum\frac{-y_i x_{i1}e^{a_i}}{(1+e^{b_i})^2}\\ \vdots \\ \sum\frac{-y_i x_{in}e^{b_i}}{(1+e^{b_i})^2}\end{bmatrix}(-N^{-1})$$

$$\nabla^2 F(\omega) = \underset{>0}{\frac{\partial F(\langle\omega,x_i\rangle(-y_i))}{\partial b}}\cdot\frac{\partial b}{\partial\omega} = \frac{1}{N}\sum_{i=1}^{N}\frac{e^{-y_i(\sum_a w_a x_{ia})}\cdot y_i^2 x_i x_i^T}{\left(1+e^{-y_i\sum w_a x_{ia}}\right)^2}$$

$$\frac{1}{N}\sum_{i=1}^{N}\underbrace{\frac{\overbrace{e^{-y_i(\sum_a w_a x_{ia})}}}{\left(1+e^{-y_i\sum w_a x_{ia}}\right)^2}}_{>0}\overbrace{y_i^2}^{\geq 0}\underbrace{x_i x_i^T}_{\geq 0}\geq 0$$

Function is convex because Hessian i PSD

$$\begin{cases} \begin{bmatrix} w_1^{(t)} \\ w_2^{(t)} \\ \vdots \\ w_d^{(t)} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix}, & \text{for } t=0,\ \vec{a} \in \mathbb{R}^d \\ \begin{bmatrix} w_1^{(t+1)} \\ w_2^{(t+1)} \\ \vdots \\ w_d^{(t+1)} \end{bmatrix} = \begin{bmatrix} w_1^t \\ w_2^t \\ \vdots \\ w_d^t \end{bmatrix} - \eta \cdot \begin{bmatrix} \sum_i^N \frac{\cancel{1}\, y_i x_{i1} e^{-y_i \sum_j^k w_j x_{ij}}}{N(1+e^{-y_i \sum w_j x_{ij}}} \\ \vdots \\ \sum_i^N \frac{y_i x_{id} e^{-y_i \sum w_j x_{ij}}}{N(1+e^{-y_i \sum w_j x_{ij}}} \end{bmatrix} & \text{else} \end{cases}$$

4.

a)

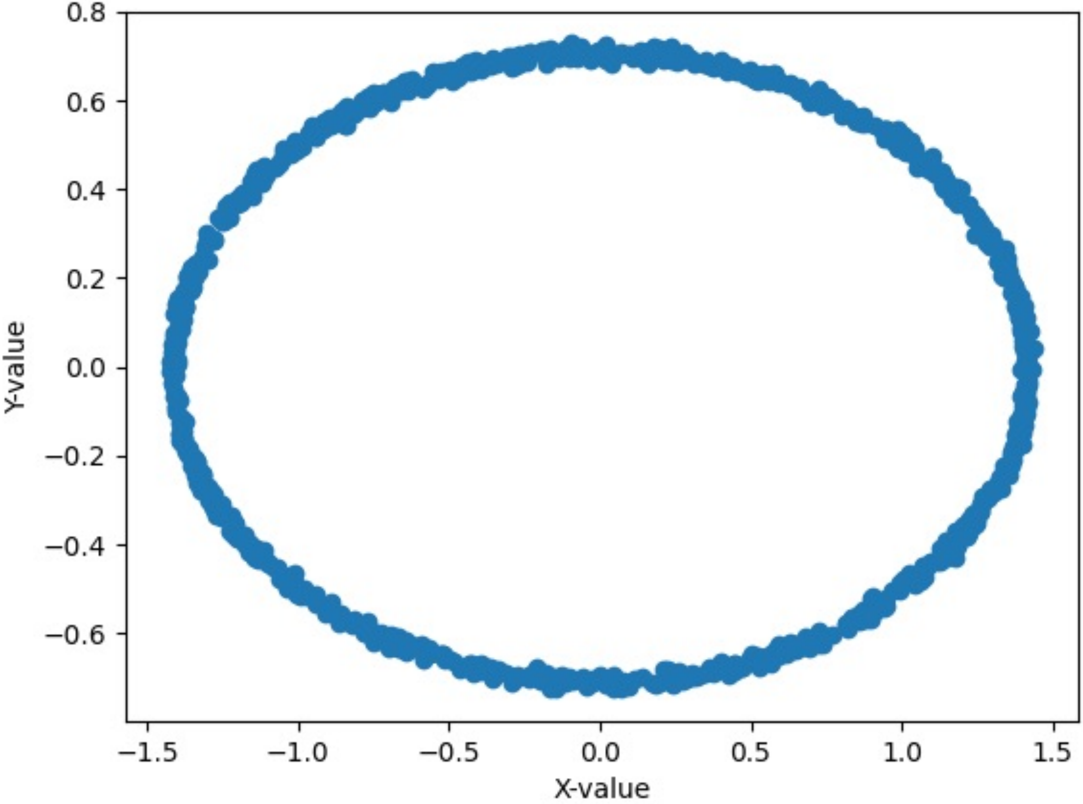
$$\|Aa-b\|_2^2 = (Aa-b)^T(Aa-b)$$

$$= \sum_{i=1}^{N} (A_{i1}a_1 + A_{i2} \cdot a_2 - b_i)^2$$

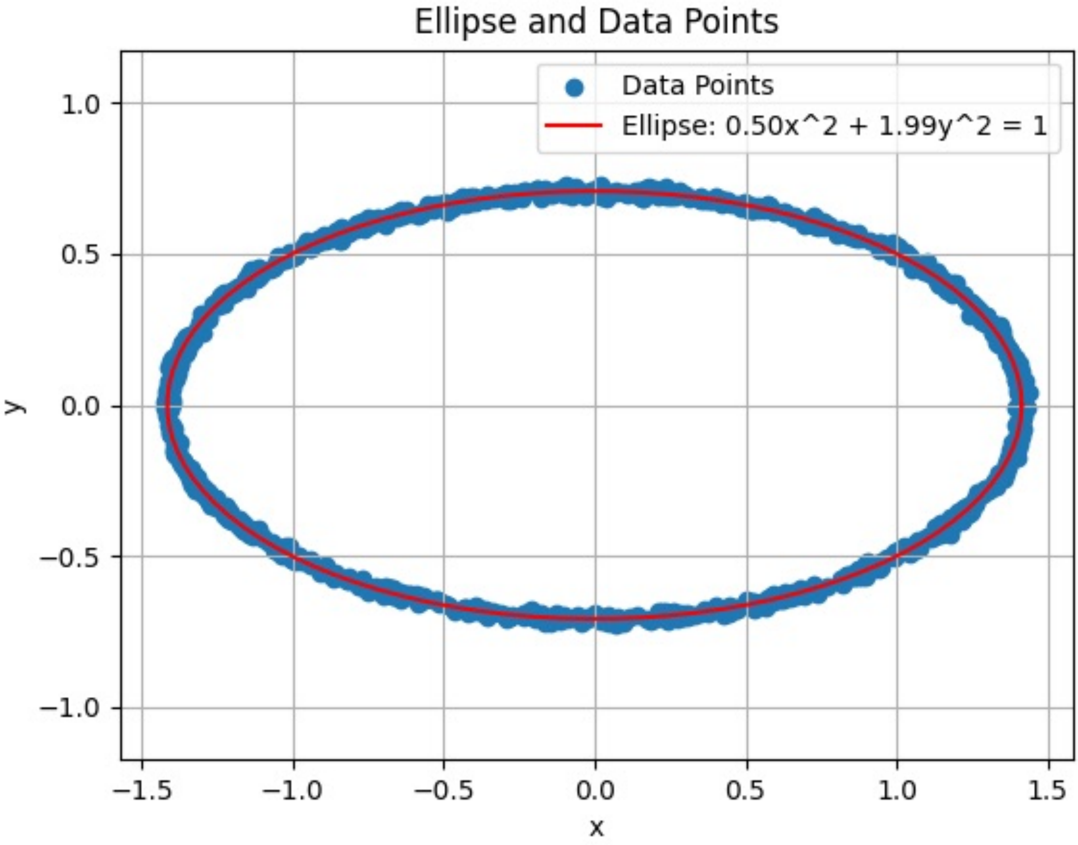
$$\sum_{i=1}^{N} (x_i^2 a_1 + y_i^2 a_2 - 1)^2 = \sum_{i=1}^{N} (A_{i1}a_1 + A_{i2} \cdot a_2 - b_i)^2$$

$$\Rightarrow A = \begin{bmatrix} x_1^2 & y_1^2 \\ x_2^2 & y_2^2 \\ \vdots & \\ x_n^2 & y_n^2 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

```
import pandas as pd
import matplotlib.pyplot as plt

csv_file_path = 'HW2_ellipse.csv'

df = pd.read_csv(csv_file_path)
df.columns = ['x', 'y']
x = df['x']
y = df['y']

plt.scatter(x,y)
plt.xlabel('X-value')
plt.ylabel('Y-value')

plt.show()
```


0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Y-value
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
X-value

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


def gradient(A:np.array, b:np.array, x:np.array):
    sum1 = 0
    sum2 = 0
    for i in range(999):
        inner_term = (A[i][0] * x[0] + A[i][1] * x[1] - 1)
        sum1 += 2 * inner_term * A[i][0]
        sum2 += 2 * inner_term * A[i][1]
    return np.array([sum1 / 999, sum2 / 999])
    # return 2 * np.dot((np.dot(A, x) - b).T, A)

def learning_rate(A:np.array):
    # 0.012774369968774082
    return abs(0.5 / np.linalg.norm(A))

def get_A(file='HW2_ellipse.csv'):
    df = pd.read_csv(file)
    df.columns = ['x', 'y']
    x = df['x']
    y = df['y']
    x_squared = x ** 2
    y_squared = y ** 2
    # Create a matrix with x_squared as column one and y_squared as column two
    A = np.column_stack((x_squared, y_squared))
    return A

def print_result(A:np.array, b:np.array, x:np.array, dim:int=999):
    print("\nX vÃ¤rde : ", x, "\n")
    print("Function-value : ", function(A, b, x), "\n")

def function(A:np.array, b:np.array, x:np.array, dim:int=999):
    sum = 0
    for i in range(dim):
        sum += (A[i][0] * x[0] + A[i][1] * x[1] - 1) ** 2
    return sum

def plot_ellipse(x: np.array, file:str='HW2_ellipse.csv'):
    # Ensure a has two elements
    # Scatter plot from the file
    df = pd.read_csv(file)
    df.columns = ['x', 'y']
    scatter_x = df['x']
    scatter_y = df['y']
    plt.scatter(scatter_x, scatter_y, label='Data Points')

    assert len(x) == 2, "Parameter array must have exactly two elements."

    a1, a2 = x

    # Generate points for the ellipse
    theta = np.linspace(0, 2 * np.pi, 100)
    ellipse_x = np.cos(theta) / np.sqrt(a1)
    ellipse_y = np.sin(theta) / np.sqrt(a2)

    # Plot the ellipse
    # plt.figure()
    plt.plot(ellipse_x, ellipse_y, color='red', label=f'Ellipse: {a1:.2f}x^2 + {a2:.2f}y^2 = 1')


    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Ellipse and Data Points')
    plt.legend()
    plt.axis('equal')
    plt.grid(True)
    plt.show()

def main():
    dim = 999
    x = np.array([random.uniform(0, 1) for _ in range(2)])
    A = get_A()
    b = np.array([1 for _ in range(dim)])
    stopping_criteria = 1e-5

    while np.linalg.norm(gradient(A, b, x)) > stopping_criteria:
        x -= learning_rate(A) * gradient(A, b, x)
    print_result(A, b, x)
    plot_ellipse(x)



if __name__ == "__main__":
    main()
```


Ellipse and Data Points
Data Points
Ellipse: 0.50x^2 + 1.99y^2 = 1
y
x
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

4 d)

Yes I think I found the optimal solution for f(a).

---

a = [0.50012765 1.99460575]

F(a) = 0.46407693961047425

Given that N=999 and the function value was low I think I found an optimal solution. Adding 999 squared sums of two squared numbers (two numbers that are even multiplied with a1 and a2) can easily become a huge number. Our x and y values were in this case relatively small, but the a-terms could easily have made f(a) a huge number if they would have been updated in the ascent direction during gradient descent. Given my small f(a) and the number N=999 I think I fit the data well.

Given the convexity of the function we know that the function is monotonically non-decreasing. This means that there could theoretically be another solution that is equally optimal. This task of fitting an ellipse however has a unique solution so I Think I found the most optimal solution.

Furthermore the convexity of the function also tells us that the hessian is always greater or equal to zero. This means that if the gradient is zero we know that the function-value won't decrease more.

When I plotted my approximated ellipse next two the scatter plot it is easy to see how similar they are.

I could have got a more accurate result however if i changed my stopping criteria. My stopping criteria was that the norm of the gradient had be less than "1e-5", if I would have chosen that as a smaller value I probably would have got a slightly more optimal solution.