1) \( x_1^2 + 2x_2^2 = 1 \) furthest away from point \((4, 3)\).

We have: \( g(x) = x_1^2 + 2x_2^2 - 1 = 0 \)

\( F(x) = (x_1 - 4)^2 + 2(x_2 - 3)^2 \)

Plug \( x(\lambda) \) into \( g(x) \):

\[
\frac{3}{2 + 2\lambda} + 2\left(\frac{17}{9 + 4\lambda}\right)^2 - 1 = 0
\]

\( x_1 = 6.1288 \)

\( x_2 = 6.020 \)

2) \( 2(x_1 + 2)^2 + 3(x_2 - 2)^2 \) Start \((0, 0)\) \( d = 0.1 \)

\( x_1^{(0)} = x_2^{(0)} = 0 - 0.1(4(0 + 2)) = -0.8 \)

\[
\begin{align*}
\frac{\partial F}{\partial x_1} &= 4(x_1 + 2) \\
\frac{\partial F}{\partial x_2} &= 6(x_2 - 2)
\end{align*}
\]

3) \( \frac{29}{2} \) \( \frac{29}{2} \)

3.2 What is machine learning? What do we know, and what do we want? We have \( x = (x_1, x_2, \ldots, x_n) \) and want an function \( f(x_1, \ldots, x_n) \) such as fast as possible.

3.5 Why cannot we have 1- or 2-layer NN?

We can't have only L or NL 1 layer because sometimes we have NL problems or such L layer can't handle. We have linear problems which NL problem can't be the best or worst give the answer since we would want a linear answer.

For 2 layer

\( L - NL \quad x_1 \cdot x_n \rightarrow y = f(x_1, \ldots, x_n) \) for this functions the level sets

\( f(x_1, \ldots, x_n) = \text{constant} \)

are plane

\( L = \frac{w_1}{2} x_1 + c \)

Thus, they can't approx e.g. \( f(x_1, x_2) = x_1 x_2 \) for which level set is hyperbola.
3.8 Which activation function is used in traditional NN? How might biological neurons use activation functions?

For a general 3-layer NN:

\[ y = \frac{1}{2} w \begin{bmatrix} w_k \end{bmatrix}_k \begin{bmatrix} k_i \end{bmatrix}_i \begin{bmatrix} -x_i \end{bmatrix}_i \begin{bmatrix} \theta_0 \end{bmatrix}_0 \]

which biological neurons use:

\[ S_0(z) = \frac{1}{1+e^{-z}} \text{ sigmoid function} \]

That's why NN use some activation function.

3.11 Explain why gradient method is used for optimization.

Locally:

\[ f(x + \alpha x) \approx f(x) + \nabla f(x)^T (x + \alpha x) \]

The idea is that once we re-scale \( x \)

we should get the same formula after we apply "natural" re-scaling:

\[ S_0(x + \alpha x) \approx S_0(x) \]

Even when it is not the most accurate, it still works and saves a lot of computational operations.