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

we have: $g(x) = x_1^2 + 2x_2^2 - 1 = 0$

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

Lagrange: $F(x) + \lambda g(x) \to$ max min

$(x_1-4)^2 + 2(x_2-3)^2 + \lambda(x_1^2 + 2x_2^2 - 1) = 0$

$\frac{\partial}{\partial x_1} = 2(x_1-4) + 2\lambda x_1 \to 2x_1 - 8 + 2\lambda x_1 = 0 \to (2+2\lambda)x_1 = 8$

$x_1 = \frac{8}{2+2\lambda}$

$\frac{\partial}{\partial x_2} = 4(x_2-3) + 4\lambda x_2 \to 4x_2 - 12 + 4\lambda x_2 = 0 \to (4+4\lambda)x_2 = 12$

$x_2 = \frac{12}{4+4\lambda}$

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

$\left(\frac{8}{2+2\lambda}\right)^2 + 2\left(\frac{12}{4+4\lambda}\right)^2 - 1 = 0$

$\lambda_1 \approx -6.1263$

$\lambda_2 \approx 6.020$

AI calculator

λ_1: $x_1 = \frac{8}{2+2\lambda} \approx -0.7802$, $x_2 = \frac{12}{4+4\lambda} \approx -0.5852$

λ_2: $x_1 \approx 0.569800$, $x_2 \approx 0.427350$

furthest

2) $2(x_1+2)^2 + 3(x_2-2)^2$ start $(0,0)$ $\alpha = 0.1$

$x_1^{(1)} = x_1^{(0)} - \alpha \frac{\partial F}{\partial x_1} = 0 - 0.1(4(0+2)) = -4/5$

$\frac{\partial F}{\partial x_1} = 4(x_1+2)$

$x_2^{(1)} = x_2^{(0)} - \alpha \frac{\partial F}{\partial x_2} = 0 - 0.1(6(0-2)) = \frac{6}{5}$ $(-4/5, 6/5)$

$\frac{\partial F}{\partial x_2} = 6(x_2-2)$

3) $3\overline{)89}$ 29 (2)

3.2 What is ~~the~~ machine learning? What do we know, and what do we want?

we have $x^k = x(x_1^{(k)}, \ldots, x_n^{(k)})$ and $y^{(k)}$ $k \to$ pattern

we want a function $f(x_1, \ldots, x_n)$, s.t. for all k $f(x_1^{(k)}, \ldots, x_n^{(k)}) \approx y^{(k)}$ 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 some times we have NL problems or such L layer cannot handle, or we have a linear problem which NL problem can't be the best or won't give the answer since we would want a linear answer.

For 2 layer

L-NL

$y_k = w_{k1}x_1 + \ldots + w_{kn}x_n - w_{k0} \to$

$y = s(w_k x_1 + \ldots + w_{kn}x_n - w_0)$

For this functions the level sets $f(x_1, \ldots, x_n) = \text{constant}$ are planes

$\sum_{i=1}^{n} w_i x_i = c$

Thus, they cannot approx

e.g. $f(x_1, x_2) = x_1 x_2$ for which level sets is hyperbola

NL-L $w_1 f_1(x_1) + \ldots + w_n f_n(x_n) + w_0$

x_1 — x_n — linear combination

ex 2 input $\to y = w_1 f_1(x_1) + w_2 f_2(x_2) + w_0 = x_1 x_2$?

$\frac{\partial}{\partial x_1} = w_1 f_1(x_1) = x_2$

$\frac{\partial}{\partial x_2} = 0 = 1 \to$ Not possible

∴ So we can't use 1- or 2- layer NN as universal

3.8 Which activation function ... [illegible]

for general 3-layer NN

$$y = \sum_{k=1}^{M} w_k s_k \left(\sum_{i=1}^{N} w_{ki} x_i - w_{k0} \right) - w_0$$

which biological neuron use

$$S_0(z) = \frac{1}{1+e^{-z}} \rightarrow \text{Sigmoid function}$$

that's why NN use same activation function.

The idea is that once we re-scale x_i we should get the same formula after we apply "natural" re-scaling.

$$S_0(x - c) = T_c(S_0(x))$$

3.11 Explain why gradient method is used for optimization.

locally $f(x+\Delta x) \approx f(x) + f'(x)\Delta x = f(x_1+\Delta x_1, x_2+\Delta x_2) = f(x_1, x_2) + \frac{\partial f}{\partial x_1}\Delta x_1 = f(x_1, x_2) + \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 +$

$$\frac{\partial f}{\partial x_1}\Delta x_1 \rightarrow f(x_1, \dots, x_n + \Delta x_n) \approx f(x_1, \dots, x_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i$$

$$f(x_1, \dots, x_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i \rightarrow \min \rightarrow \sum_{i=1}^{n} \Delta x_i^2 = \delta^2 \longrightarrow \sqrt{\sum \Delta x_i^2} = \delta$$

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$f + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} + \lambda \left(\sum_{i=1}^{n} \Delta x_i^2 - \delta^2 \right) \rightarrow \min_{\Delta x_i}$$

$$\frac{\partial f}{\partial x_i} + 2\lambda \Delta x_i = 0 \qquad \Delta x_i = -\frac{1}{2\lambda}\frac{\partial f}{\partial x_i}$$

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