1. Newton's Method

a) Critical Point and Newton Update Rule

Given the second-order Taylor expansion of the empirical risk $R_{emp}(\theta)$ near θ_0:

$$R_{emp}(\theta) \approx R_{emp}(\theta_0) + (\theta - \theta_0)^T \nabla_\theta R_{emp}(\theta_0) + \frac{1}{2}(\theta - \theta_0)^T H_\theta (\theta - \theta_0)$$

where H_θ is the Hessian of R_{emp} with respect to θ evaluated at θ_0.

To find the minimum, set the gradient with respect to θ to zero:

$$\nabla_\theta R_{emp}(\theta) \approx \nabla_\theta R_{emp}(\theta_0) + H_\theta(\theta - \theta_0) = 0$$

Solving for θ:

$$H_\theta(\theta - \theta_0) = -\nabla_\theta R_{emp}(\theta_0)$$
$$\theta = \theta_0 - H_\theta^{-1} \nabla_\theta R_{emp}(\theta_0)$$

Newton's update rule:

$$\boxed{\theta_{new} = \theta_0 - H_\theta^{-1} \nabla_\theta R_{emp}(\theta_0)}$$

b) Newton's Method for $f(x) = (x-1)(x-3)$ at $x_0 = 4$

First, expand $f(x)$:

$$f(x) = (x-1)(x-3) = x^2 - 4x + 3$$

Compute first and second derivatives:

$$f'(x) = 2x - 4$$
$$f''(x) = 2$$

Newton's update rule for scalar x:

$$x_{new} = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Plug in $x_0 = 4$:

$$f'(4) = 2 \times 4 - 4 = 4$$
$$f''(4) = 2$$
$$x_{new} = 4 - \frac{4}{2} = 2$$

Newton's method jumps directly to the minimum at $x = 2$ in one step.

c) Difficulties in Neural Networks

Applying Newton's method and other second-order methods to neural networks is difficult because:

- **Computational cost:** The Hessian matrix is very large for networks with many parameters, making its computation and inversion expensive.
- **Memory requirements:** Storing the Hessian is often infeasible for large models.
- **Non-convexity:** Neural network loss surfaces are highly non-convex, so the Hessian may not be positive definite, leading to unstable updates.

2. Vanishing Gradients

a) Sigmoid Slope & Vanishing Gradients

The sigmoid function has a slope close to zero at points far from the origin (i.e., for large positive or negative x). At x_1 and x_2 (as shown in the graph), the slope is very small, meaning the gradient is nearly zero. This exemplifies the vanishing gradient problem: during backpropagation, gradients become very small, slowing or stopping learning in deep networks.

b) Forward Pass & Gradients

Network Structure:

- Input: $x = 1.5$
- Hidden neuron: weight $\theta_1 = 4$, bias $= 0$
- Output neuron: weight $\theta_2 = 1.5$, bias $= 0$
- Loss: $L = \frac{1}{2}(y_{pred} - y)^2$, with $y = 1$

i) Sigmoid Activations

Forward pass:

$$z = \theta_1 x = 4 \times 1.5 = 6$$
$$h = \sigma(z) = \frac{1}{1+e^{-6}} \approx 0.9975$$
$$o = \theta_2 h = 1.5 \times 0.9975 \approx 1.4963$$
$$y_{pred} = \sigma(o) = \frac{1}{1+e^{-1.4963}} \approx 0.8170$$

Loss:

$$L = \frac{1}{2}(0.8170 - 1)^2 \approx 0.0167$$

Gradients:

$$\frac{\partial L}{\partial y_{pred}} = y_{pred} - y = 0.8170 - 1 = -0.1830$$
$$\frac{\partial y_{pred}}{\partial o} = \sigma'(o) = y_{pred}(1 - y_{pred}) \approx 0.8170 \times 0.1830 \approx 0.1496$$
$$\frac{\partial o}{\partial \theta_2} = h \approx 0.9975$$
$$\frac{\partial L}{\partial \theta_2} = (-0.1830) \times 0.1496 \times 0.9975 \approx -0.0274$$

For θ_1:

$$\frac{\partial o}{\partial h} = \theta_2 = 1.5$$
$$\frac{\partial h}{\partial z} = \sigma'(z) = h(1-h) \approx 0.9975 \times 0.0025 \approx 0.0025$$
$$\frac{\partial z}{\partial \theta_1} = x = 1.5$$
$$\frac{\partial L}{\partial \theta_1} = (-0.1830) \times 0.1496 \times 1.5 \times 0.0025 \times 1.5 \approx -0.0002$$

Interpretation: The gradient for θ_1 is extremely small due to the sigmoid's saturation at large input values ($z = 6$), exemplifying vanishing gradients.

ii) ReLU Activations

Forward pass:

$$\begin{aligned} z &= 6 \\ h &= \max(0, z) = 6 \\ o &= \theta_2 h = 1.5 \times 6 = 9 \\ y_{pred} &= \max(0, o) = 9 \end{aligned}$$

Loss:

$$L = \frac{1}{2}(9-1)^2 = \frac{1}{2} \times 64 = 32$$

Gradients:

$$\begin{aligned} \frac{\partial L}{\partial y_{pred}} &= y_{pred} - y = 8 \\ \frac{\partial y_{pred}}{\partial o} &= 1 \quad (\text{since } o > 0) \\ \frac{\partial o}{\partial \theta_2} &= h = 6 \\ \frac{\partial L}{\partial \theta_2} &= 8 \times 1 \times 6 = 48 \end{aligned}$$

For θ_1:

$$\begin{aligned} \frac{\partial o}{\partial h} &= \theta_2 = 1.5 \\ \frac{\partial h}{\partial z} &= 1 \quad (\text{since } z > 0) \\ \frac{\partial z}{\partial \theta_1} &= x = 1.5 \\ \frac{\partial L}{\partial \theta_1} &= 8 \times 1 \times 1.5 \times 1 \times 1.5 = 27 \end{aligned}$$

Interpretation: The gradients are much larger and do not vanish, showing that ReLU avoids the vanishing gradient problem.

Interpretation & Geometric Explanation

- **Sigmoid/tanh:** Gradients vanish for large input values because the slope of the activation function approaches zero.
- **ReLU:** The gradient is constant (1) for positive inputs, so gradients do not vanish.

- **Tanh:** Like sigmoid, tanh saturates for large positive/negative inputs, causing vanishing gradients.

Geometric explanation: For sigmoid/tanh, the function flattens out at the extremes, so the derivative (slope) is near zero. For ReLU, the function is linear for positive inputs, so the slope remains constant.