1.

$$f(y_i;\boldsymbol{\beta}) = \mu_i^{y_i}(1-\mu_i)^{1-y_i}, \quad y_i \in \{0,1\}, \quad \text{where } \mu_i = \Phi(\mathbf{x}_i^\top \boldsymbol{\beta}).$$

So each term can be written as

$$P(y_i \mid x_i, \boldsymbol{\beta}) = \big(\Phi(x_i^\top \boldsymbol{\beta})\big)^{y_i}\big(1-\Phi(x_i^\top \boldsymbol{\beta})\big)^{1-y_i}$$

The likelihood function is

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(y_i \mid x_i, \boldsymbol{\beta}), \qquad y_i \in \{0,1\}$$

Let

$$P(y_i = 1 \mid x_i, \boldsymbol{\beta}) = \Phi(x_i^\top \boldsymbol{\beta}), \qquad P(y_i = 0 \mid x_i, \boldsymbol{\beta}) = 1 - \Phi(x_i^\top \boldsymbol{\beta}), \qquad y_i \in \{0,1\}.$$

The Product likelihood:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(y_i \mid x_i, \boldsymbol{\beta})$$

$$\Rightarrow \quad L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \big(\Phi(x_i^\top \boldsymbol{\beta})\big)^{y_i}\big(1-\Phi(x_i^\top \boldsymbol{\beta})\big)^{1-y_i}$$

$$\Rightarrow \quad \log L(\boldsymbol{\beta}) = \log \prod_{i=1}^{n} \Big(\big(\Phi(x_i^\top \boldsymbol{\beta})\big)^{y_i}\big(1-\Phi(x_i^\top \boldsymbol{\beta})\big)^{1-y_i}\Big) \qquad \text{(taking log)}$$

$$\Rightarrow \quad \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log \Big(\big(\Phi(x_i^\top \boldsymbol{\beta})\big)^{y_i}\big(1-\Phi(x_i^\top \boldsymbol{\beta})\big)^{1-y_i}\Big) \qquad \text{(since } \log \prod_i a_i = \sum_i \log a_i\text{)}$$

$$\Rightarrow \quad \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \Big[\log\big(\Phi(x_i^\top \boldsymbol{\beta})^{y_i}\big) + \log\big((1-\Phi(x_i^\top \boldsymbol{\beta}))^{1-y_i}\big)\Big] \qquad \text{(since } \log(ab) = \log(a) + \log(b)\text{)}$$

$$\Rightarrow \quad \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \Big[y_i \log \Phi(x_i^\top \boldsymbol{\beta}) + (1-y_i) \log\big(1-\Phi(x_i^\top \boldsymbol{\beta})\big)\Big] \qquad \text{(since } \log(a^b) = b\log(a)\text{).}$$

Sanity check using $y_i \in \{0,1\}$:

- If $y_i = 1$, the term becomes $\log \Phi(\mathbf{x}_i^\top \boldsymbol{\beta})$
- If $y_i = 0$, the term becomes $\log\big(1-\Phi(\mathbf{x}_i^\top \boldsymbol{\beta})\big)$

The log-likelihood function for probit regression represents the logarithm of the probability of observing the entire dataset given parameters $\boldsymbol{\beta}$. We maximize this to find the best-fitting model:

$$\log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \big[y_i \log \Phi(\mathbf{x}_i^\top \boldsymbol{\beta}) + (1-y_i) \log\big(1-\Phi(\mathbf{x}_i^\top \boldsymbol{\beta})\big)\big]$$

The predicted probability $\mu(\mathbf{x})$ is obtained by transforming the linear predictor $\boldsymbol{\beta}^\top \mathbf{x}$ through the standard normal CDF Φ. This maps any real value to a probability between 0 and 1:

$$\mu(\mathbf{x}) = \Phi(\boldsymbol{\beta}^\top \mathbf{x}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\boldsymbol{\beta}^\top \mathbf{x}} \exp\left(-\frac{u^2}{2}\right) du$$

The derivative of the standard normal CDF with respect to its argument is the standard normal PDF $\phi(z)$.

$$\frac{d}{dz}\Phi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) = \phi(z)$$

Applying this to the predicted probability function, the rate of change of $\mu(\mathbf{x})$ with respect to the linear predictor is:

$$\frac{\partial}{\partial(\boldsymbol{\beta}^\top \mathbf{x})}\mu(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x})^2}{2}\right)$$

Let $g_i(\boldsymbol{\beta})$ be the contribution of the i-th observation to the log-likelihood.To compute the gradient efficiently, we first consider the log-likelihood contribution from a single observation i. This allows to derive the gradient term-by-term:

$$g_i(\boldsymbol{\beta}) = y_i \log\left[\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)\right] + (1 - y_i) \log\left[1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)\right]$$

We compute the gradient of the term that activates when the outcome is positive ($y_i = 1$). This uses the chain rule three times: for the logarithm, the CDF, and the linear predictor:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\beta}} \left\{ y_i \log[\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)] \right\} &= y_i \cdot \frac{1}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \frac{\partial}{\partial \boldsymbol{\beta}} \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \\
&\quad \text{(Step 1: Apply chain rule; derivative of } \log(u) \text{ is } \frac{1}{u}) \\
&= y_i \cdot \frac{1}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \mathbf{x}_i \\
&\quad \text{(Step 2: Use } \frac{d\Phi(z)}{dz} = \phi(z) \text{ and } \frac{\partial(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\partial \boldsymbol{\beta}} = \mathbf{x}_i) \\
&= y_i \cdot \frac{1}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \right) \cdot \mathbf{x}_i \\
&\quad \text{(Step 3: Substitute the explicit form of the standard normal PDF)}
\end{aligned}$$

Similarly, we compute the gradient of the term that activates when the outcome is negative ($y_i = 0$). The key difference is the negative sign from differentiating $(1 - \Phi)$:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\beta}} \left\{ (1 - y_i) \log[1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)] \right\} &= (1 - y_i) \cdot \frac{1}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \frac{\partial}{\partial \boldsymbol{\beta}} (1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)) \\
&\quad \text{(Step 1: Apply chain rule to } \log(1 - u)) \\
&= (1 - y_i) \cdot \frac{1}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot (-\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)) \cdot \mathbf{x}_i \\
&\quad \text{(Step 2: Derivative of } (1 - \Phi) \text{ is } -\phi \text{; include gradient of linear predictor)} \\
&= (1 - y_i) \cdot \frac{1}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \left(-\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \right) \cdot \mathbf{x}_i \\
&\quad \text{(Step 3: Substitute the explicit form of the standard normal PDF)} \\
&= -(1 - y_i) \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \cdot \mathbf{x}_i \\
&\quad \text{(Step 4: Rearrange to show the PDF in the numerator)}
\end{aligned}$$

Combining both partial derivatives gives the complete gradient contribution from observation i:

$$\frac{\partial g_i}{\partial \boldsymbol{\beta}} = \left(y_i \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - (1 - y_i) \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right) \mathbf{x}_i$$

Starting with the gradient contributions from all n observations, we sum the individual derivatives we computed earlier:

$$\nabla_{\boldsymbol{\beta}} \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - (1 - y_i) \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right] \mathbf{x}_i$$

(This is the sum of all $\dfrac{\partial g_i}{\partial \boldsymbol{\beta}}$ terms)

Both terms in the brackets contain the same normal PDF $\phi(\boldsymbol{\beta}^\top \mathbf{x}_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)$. We factor this out:

$$= \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \left[\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1 - y_i}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right] \mathbf{x}_i$$

(Factoring out the standard normal PDF $\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)$)

To simplify the expression in brackets, we find a common denominator and combine the fractions:

$$= \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \cdot \frac{y_i(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)) - (1 - y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \cdot \mathbf{x}_i$$

(Common denominator is $\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))$)

Expanding the numerator:

$$\begin{aligned}
&\text{Numerator} = y_i(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)) - (1 - y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \\
&= y_i - y_i\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) + y_i\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \\
&= y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \\
&\quad \text{(The } y_i\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \text{ terms cancel)}
\end{aligned}$$

Therefore:

$$= \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \cdot \mathbf{x}_i$$

The gradient of the log-likelihood is:

$$\nabla_{\boldsymbol{\beta}} \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \cdot \mathbf{x}_i$$

where $\phi(\cdot)$ is the standard normal PDF and $\Phi(\cdot)$ is the standard normal CDF.

To find the Hessian matrix, we take the derivative of the gradient with respect to $\boldsymbol{\beta}$. Since each term in the gradient has the form [scalar] $\cdot \mathbf{x}_i$, the Hessian will involve outer products $\mathbf{x}_i \mathbf{x}_i^\top$:

$$\nabla^2_{\boldsymbol{\beta}} \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(\frac{\partial}{\partial \boldsymbol{\beta}} \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1 - y_i}{1 - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right) \right] \right) \mathbf{x}_i^\top$$

We need to compute the derivative of the product of the PDF term and the fraction term.
First, we compute how the normal PDF changes with respect to its argument. Using the chain rule on the exponential:

$$\begin{aligned}
\frac{d}{d(\boldsymbol{\beta}^\top \mathbf{x}_i)} \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \right] &= -(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right) \\
&= -(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \phi(\boldsymbol{\beta}^\top \mathbf{x}_i)
\end{aligned}$$

Using the quotient rule $\frac{d}{dz}\left(\frac{u}{v}\right) = -\frac{u \cdot v'}{v^2}$ when u is constant and v depends on z:

$$\begin{aligned}\frac{d}{d(\boldsymbol{\beta}^\top \mathbf{x}_i)}\left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right) &= -y_i \cdot \frac{\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} \\ &= -y_i \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}\end{aligned}$$

Similarly, for the second term, but now the derivative of the denominator $(1 - \Phi)$ is $-\phi$, giving a positive result:

$$\begin{aligned}\frac{d}{d(\boldsymbol{\beta}^\top \mathbf{x}_i)}\left(\frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right) &= (1-y_i) \cdot \frac{\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} \\ &= (1-y_i) \cdot \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\end{aligned}$$

Adding the derivatives of both fraction terms and factoring out the common PDF:

$$\begin{aligned}&\frac{d}{d(\boldsymbol{\beta}^\top \mathbf{x}_i)}\left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right) \\ &\quad = -y_i \cdot \frac{\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + (1-y_i) \cdot \frac{\phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} \\ &\qquad \text{(Substituting the derivatives computed above)} \\ &\quad = \phi(\boldsymbol{\beta}^\top \mathbf{x}_i)\left[-\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + \frac{1-y_i}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\right] \\ &\qquad \text{(Factoring out the common normal PDF)} \\ &\quad = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}\right)\left[-\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + \frac{1-y_i}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\right] \\ &\qquad \text{(Writing out the PDF explicitly)}\end{aligned}$$

To find the second derivative, we apply the product rule to the product of the PDF and the fraction terms. Recall that for a product $f \cdot g$, we have $(f \cdot g)' = f' \cdot g + f \cdot g'$:

$$\begin{aligned}&\frac{d}{d(\boldsymbol{\beta}^\top \mathbf{x}_i)}\left[\phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right)\right] \\ &\qquad \text{(Setting up the product rule for [PDF] } \times \text{ [fraction terms])}\end{aligned}$$

The product rule gives two terms: the derivative of the PDF times the original fractions, plus the PDF times the derivative of the fractions:

$$\begin{aligned}&= \left(-(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}}\right)\left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right) \\ &\quad + \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}}\left[-\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + \frac{1-y_i}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\right] \\ &\qquad \text{(First term: derivative of } \phi\text{, which is } \phi'(z) = -z\phi(z)\text{)} \\ &\qquad \text{(Second term: } \phi \text{ times the derivative of fractions from the previous calculation)}\end{aligned}$$

We factor out $\phi(\boldsymbol{\beta}^\top \mathbf{x}_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}}$ from both terms:

$$\begin{aligned}&= \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}}\left[-(\boldsymbol{\beta}^\top \mathbf{x}_i)\left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}\right)\right. \\ &\qquad \left. + \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}}\left(-\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + \frac{1-y_i}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\right)\right] \\ &\qquad \text{(Factoring out the first PDF; note the second term has } \phi^2\text{)}\end{aligned}$$

We convert the fractions to common denominators. For the first bracket, we use the fact that:

$$\frac{y_i}{\Phi} - \frac{1-y_i}{1-\Phi} = \frac{y_i(1-\Phi) - (1-y_i)\Phi}{\Phi(1-\Phi)} = \frac{y_i - \Phi}{\Phi(1-\Phi)}$$

For the second bracket, we find a common denominator of $\Phi^2(1-\Phi)^2$:

$$= \phi(\boldsymbol{\beta}^\top \mathbf{x}_i)\Bigg[- (\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} - \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2 - (1-y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2}\Bigg]$$

(Simplifying fractions; note the negative sign in the second term)

The Hessian matrix is the matrix of second partial derivatives. We start by differentiating the gradient with respect to $\boldsymbol{\beta}$:

$$\nabla^2_{\boldsymbol{\beta}} \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(\frac{\partial}{\partial \boldsymbol{\beta}} \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2} \right) \left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right) \right] \right) \mathbf{x}_i^\top$$

(The outer product $\mathbf{x}_i^\top$ comes from differentiating the $\mathbf{x}_i$ factor in the gradient)

Using the product rule calculation from the previous work, we substitute the derivative and include the outer product $\mathbf{x}_i \mathbf{x}_i^\top$:

$$= \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}} \Bigg[- (\boldsymbol{\beta}^\top \mathbf{x}_i) \left(\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} - \frac{1-y_i}{1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)} \right) + \frac{1}{\sqrt{2\pi}} e^{-\frac{(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{2}} \left(-\frac{y_i}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2} + \frac{1-y_i}{(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} \right) \Bigg] \mathbf{x}_i \mathbf{x}_i^\top$$

(Note: $\mathbf{x}_i \mathbf{x}_i^\top$ is a $p \times p$ matrix where p is the dimension of $\boldsymbol{\beta}$)

We factor out the negative sign and rearrange terms to highlight the structure. Note that we've corrected the sign in the numerator:

$$= -\sum_{i=1}^{n} \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \Bigg[\phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2 + (1-y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} + (\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \Bigg] \mathbf{x}_i \mathbf{x}_i^\top$$

(Factoring out $-\phi$ and using simplified fractions from earlier)

Distributing the negative sign to both terms inside the brackets gives the final expression:

$$= \sum_{i=1}^{n} \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \Bigg[- \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2 + (1-y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} - (\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \Bigg] \mathbf{x}_i \mathbf{x}_i^\top$$

$$\nabla^2_{\boldsymbol{\beta}} \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \left[- \phi(\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2 + (1-y_i)\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)^2(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))^2} - (\boldsymbol{\beta}^\top \mathbf{x}_i) \cdot \frac{y_i - \Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)}{\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i)(1-\Phi(\boldsymbol{\beta}^\top \mathbf{x}_i))} \right] \mathbf{x}_i \mathbf{x}_i^\top$$

```python
def newton_raphson(model, tol=1e-5, max_iter=1000, display=True):
    i = 0
    error = 100
    if display:
        header = f'{"Iteration_k":<13}{"Log-likelihood":<16}{"θ":<60}'
```

```
        print(header)
        print("-" * len(header))

    while np.any(error > tol) and i < max_iter:
        H, G = model.H(), model.G()
        β_new = model.β - (np.linalg.inv(H) @ G)
        error = β_new - model.β
        model.β = β_new
        if display:
            β_list = [f'{t:.3}' for t in list(model.β.flatten())]
            update = f'{i:<13}{model.logL():<16.8}{β_list}'
            print(update)

        i += 1

    print(f'Number of iterations: {i}')
    print(f'β_hat = {model.β.flatten()}')
    return model.β.flatten()
```

```
class ProbitModel:
    def __init__(self, X, y, beta_init):
        self.X = np.asarray(X, dtype=float)
        self.y = np.asarray(y, dtype=float).reshape(-1)
        self.β= np.asarray(beta_init, dtype=float).reshape(-1)

    def logL(self) -> float:
        z = self.X @ self.β
        return float(np.sum(self.y * norm.logcdf(z) + (1 - self.y) * norm.logsf(z)))

    def G(self) -> np.ndarray:
        z   = self.X @ self.β
        Phi = norm.cdf(z)
        phi = norm.pdf(z)
        eps = np.finfo(float).eps
        denom = np.clip(Phi * (1 - Phi), eps, None)
        w = phi * (self.y - Phi) / denom
        return self.X.T @ w

    def H(self) -> np.ndarray:
        z   = self.X @ self.β
        Phi = norm.cdf(z)
        phi = norm.pdf(z)
        eps  = np.finfo(float).eps
        PhiS = np.clip(Phi, eps, 1 - eps)
        OMS  = 1 - PhiS
        A_num = self.y * (OMS**2) + (1 - self.y) * (PhiS**2)
        A_den = (PhiS**2) * (OMS**2)
        A = A_num / A_den
        B = (self.y - PhiS) / (PhiS * OMS)
        coeff = phi * ( - phi * A - z * B )
        return self.X.T @ (coeff[:, None] * self.X)
```

2.

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 1 & 1 \\ 1 & 4 & 3 \\ 1 & 5 & 6 \\ 1 & 3 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \boldsymbol{\beta}_{(0)} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

Based on the given dataset and the initial value of $\boldsymbol{\beta}_{(0)} = (0.1, 0.1, 0.1)^\top$, we applied the Newton–Raphson algorithm to maximize the probit log-likelihood function. The iteration process converged after 5 steps, yielding the following maximum likelihood estimate:

$$\hat{\boldsymbol{\beta}} = \begin{bmatrix} -1.5463 \\ 0.7778 \\ -0.0971 \end{bmatrix}$$

```
Iteration_k  Log-likelihood  θ
-------------------------------------------------------------
0            -2.3796884     ['-1.34', '0.775', '-0.157']
1            -2.3687526     ['-1.53', '0.775', '-0.0981']
2            -2.3687294     ['-1.55', '0.778', '-0.0971']
3            -2.3687294     ['-1.55', '0.778', '-0.0971']
4            -2.3687294     ['-1.55', '0.778', '-0.0971']
Number of iterations: 5
β_hat = [-1.54625858  0.77778952 -0.09709757]
```

```
Optimization terminated successfully.
         Current function value: 0.473746
         Iterations 6
                          Probit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                    5
Model:                         Probit   Df Residuals:                        2
Method:                           MLE   Df Model:                            2
Date:                Mon, 29 Sep 2025   Pseudo R-squ.:                  0.2961
Time:                        19:55:22   Log-Likelihood:                -2.3687
converged:                       True   LL-Null:                       -3.3651
Covariance Type:            nonrobust   LLR p-value:                    0.3692
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.5463      1.866     -0.829      0.407      -5.204       2.111
x1             0.7778      0.788      0.986      0.324      -0.768       2.323
x2            -0.0971      0.590     -0.165      0.869      -1.254       1.060
==============================================================================

Estimated coefficients (β_hat):
[-1.54625858  0.77778952 -0.09709757]
```

```python
X = np.array([
    [1, 2, 4],
    [1, 1, 1],
    [1, 4, 3],
    [1, 5, 6],
    [1, 3, 5],
], dtype=float)

y = np.array([1, 0, 1, 1, 0], dtype=float)
beta0 = np.array([0.1, 0.1, 0.1], dtype=float)

model = ProbitModel(X, y, beta0)
beta_hat = newton_raphson(model, display=True)
```

```python
import numpy as np
import pandas as pd
from statsmodels.discrete.discrete_model import Probit
import statsmodels.api as sm

X = np.array([
    [1, 2, 4],
    [1, 1, 1],
```

```
    [1, 4, 3],
    [1, 5, 6],
    [1, 3, 5],
], dtype=float)

y = np.array([1, 0, 1, 1, 0], dtype=float)

model = Probit(y, X)
result = model.fit(disp=True)

print(result.summary())
print("\nEstimated coefficients (beta_hat):")
print(result.params)
```

3.

```
Iteration_k  Log-likelihood  θ
-------------------------------------------------------------
0            -5603.7745      ['1.78', '0.74', '0.176']
1            -5239.8158      ['1.58', '0.741', '0.155']
2            -5234.0225      ['1.56', '0.742', '0.148']
3            -5234.0204      ['1.56', '0.742', '0.148']
4            -5234.0204      ['1.56', '0.742', '0.148']
Number of iterations: 5
β_hat = [1.55904334 0.74199832 0.14767199]
```

```
Intercept: 1.5590425134556154
Coefficients: [0.74199947 0.14767126]
Model Score: 0.39440825681225433
```

```
Optimization terminated successfully.
         Current function value: 5.234020
         Iterations 6
                          Poisson Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1000
Model:                        Poisson   Df Residuals:                      997
Method:                           MLE   Df Model:                            2
Date:                Mon, 29 Sep 2025   Pseudo R-squ.:                  0.3092
Time:                        20:59:09   Log-Likelihood:                -5234.0
converged:                       True   LL-Null:                       -7577.0
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.5590      0.026     60.033      0.000       1.508       1.610
x1             0.7420      0.011     65.799      0.000       0.720       0.764
x2             0.1477      0.010     14.078      0.000       0.127       0.168
```

The Newton-Raphson algorithm shoes good convergence behavior, achieving optimal parameter estimates in only 5 iterations with $\hat{\boldsymbol{\beta}} = [1.55904334, 0.74199832, 0.14767199]^T$. The convergence pattern shows rapid improvement, with the log-likelihood value moving from -5603.7745 at initialization to -5239.8158 after the first iteration, then fine-tuning to -5234.0204 by iteration 3 where practical convergence was achieved. The sklearn implementation produced identical results with intercept 1.55904251 and coefficients $[0.74199947, 0.14767126]$, differing by less than 10^{-6} from Newton-Raphson, while achieving a D^2 model score of 0.3944 indicating that 39.4% of the deviance is explained by the model.

The statsmodels output, converging in 6 iterations (one more than Newton-Raphson's 5) to an identical log-likelihood of -5234.0, validates the Newton-Raphson implementation with parameter estimates matching to within numerical precision: statsmodels reports $\beta_0 = 1.5590$, $\beta_1 = 0.7420$, and $\beta_2 = 0.1477$, which agree with Newton-Raphson's estimates to 4 decimal places.

The maximum absolute difference between all three methods is less than 1.15×10^{-6}, confirming that Newton-Raphson, sklearn, and statsmodels have converged to the same global optimum. While sklearn reports a D^2 model score of 0.3944 and statsmodels provides a pseudo R-squared of 0.3092, both indicate strong model fit, and all three implementations achieve the identical maximized log-likelihood of -5234.02, demonstrating agreement in the optimization solution using different algorithms.

```python
from scipy.special import gammaln

class PoissonRegression:

    def __init__(self, y, X, β):
        self.X = X
        self.n, self.k = X.shape
        self.y = y.reshape(self.n,1)
        self.β = β.reshape(self.k,1)

    def μ(self):
        return np.exp(self.X @ self.β)

```

```
    def logL(self):
        y = self.y
        μ = self.μ()
        return np.sum(y * np.log(μ) - μ - gammaln(y + 1))

    def G(self):
        X = self.X
        y = self.y
        μ = self.μ()
        return X.T @ (y - μ)

    def H(self):
        X = self.X
        μ = self.μ()
        return -X.T @ (μ * X)
```

```
X = df[df.columns.drop('y')]
y = df['y']
Xc=sm.add_constant(X, prepend=True)

X_m = Xc.values
y_m = y.values
```

```
init_β = np.array([np.log(y_m.mean()), 0.0, 0.0])
poi = PoissonRegression(y_m, X_m, init_β)
β_hat = newton_raphson(poi)
```

```
model = PoissonRegressor(alpha=0, max_iter=1000)
model.fit(X, y)
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
print(f"Model Score: {model.score(X, y)}")
```

```
Xc=sm.add_constant(X, prepend=True)
mod_p = sm.Poisson(y, Xc)
res_p = mod_p.fit()
print(res_p.summary())
```