Logistic Regression:
A Probabilistic Linear Classifier

By: Dominic Sciarrino (MAT616)

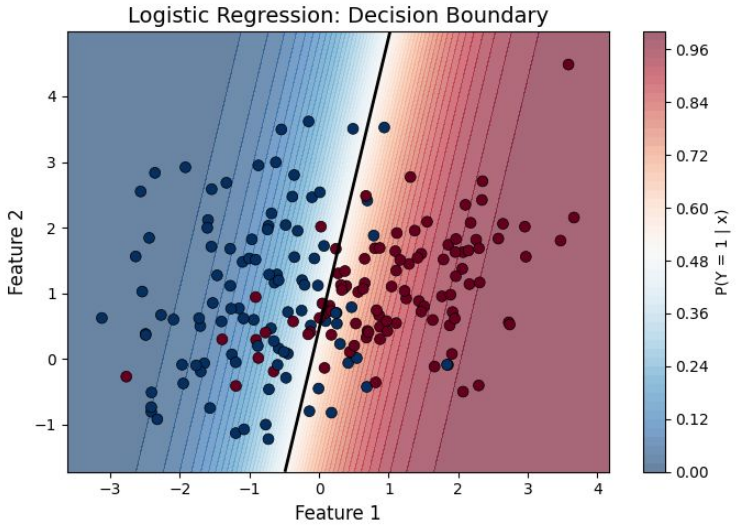
An Introduction to Logistic Regression

- Logistic regression is a **supervised learning algorithm** used for **classification**.

- It models the **probability** that an observation belongs to a given **class**, conditional on its features.

- It is a linear model at its core: the decision boundary it learns is a hyperplane in the feature space.

$$P(Y = 1 \mid \mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\beta})$$



From Linear Regression to Classification

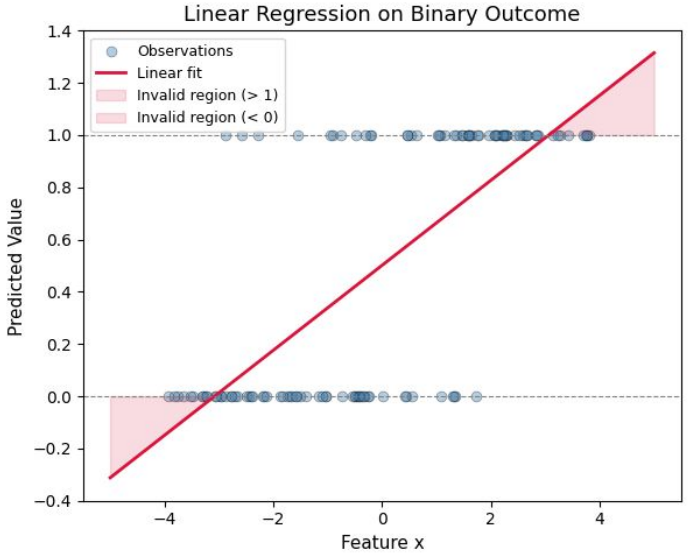
Linear Model:

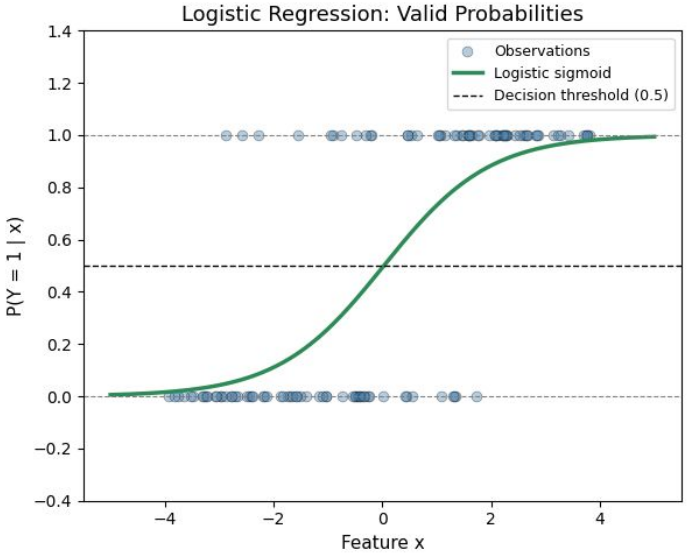
$$\hat{y} = \mathbf{x}^T \boldsymbol{\beta}, \quad \hat{y} \in (-\infty, +\infty)$$

Classification Model:

$$\hat{y} = \begin{cases} 1 & \text{if } P(Y = 1 \mid \mathbf{x}) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$P(Y = 1 \mid \mathbf{x}) \in [0, 1]$$





Important Assumptions for Logistic Regression

1. Independence of Observations

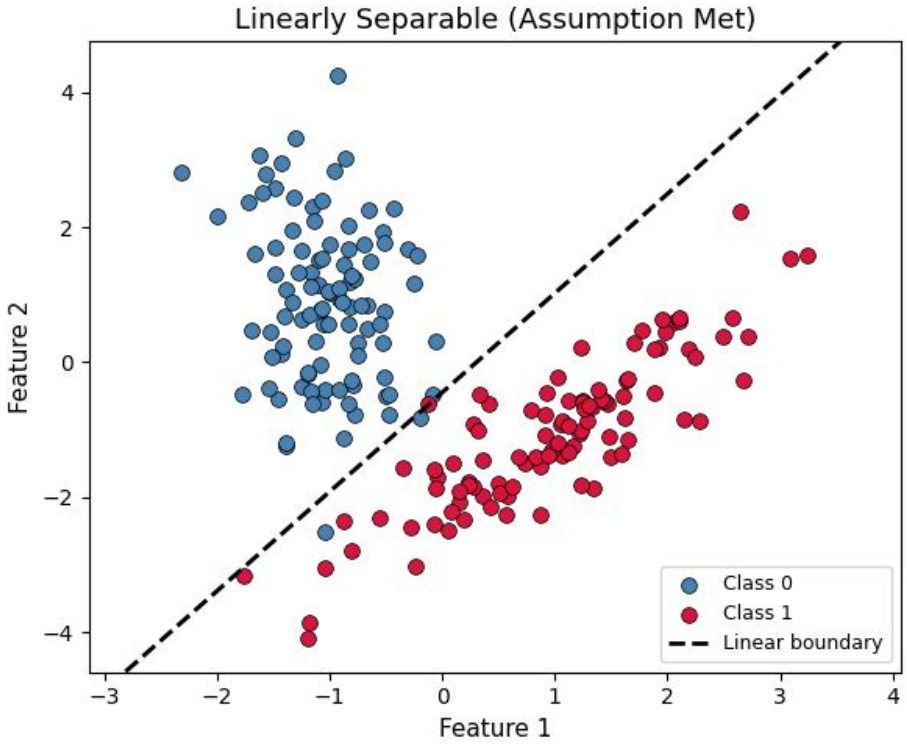
$$P(Y_1, Y_2, \ldots, Y_n) = \prod_{i=1}^{n} P(Y_i)$$

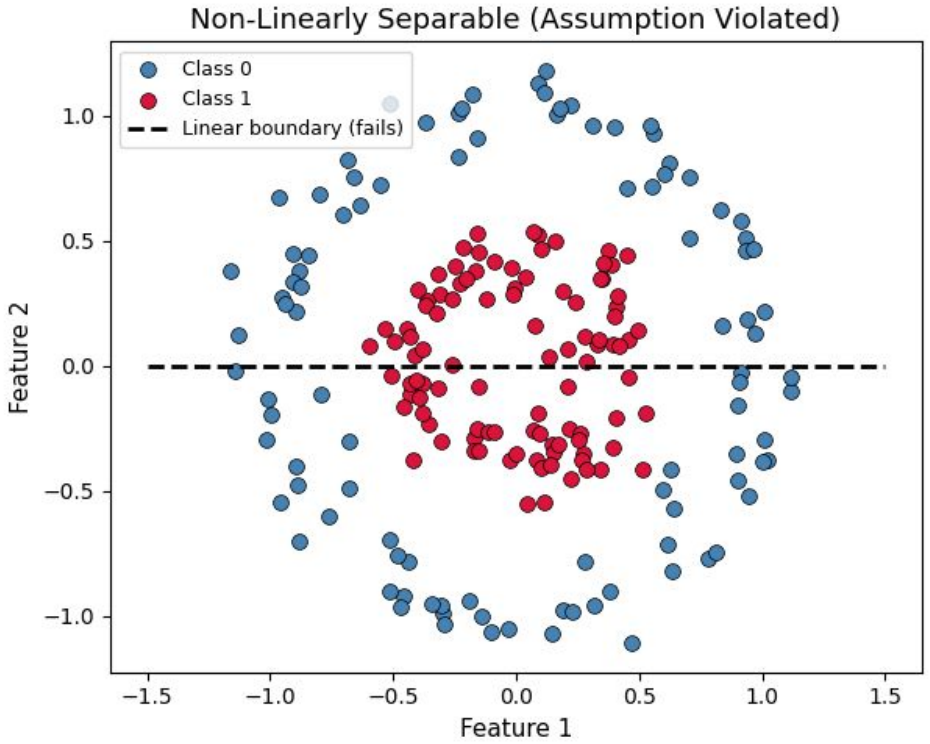
2. Linear Decision Boundary

$$\log \frac{p_i}{1 - p_i} = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} = \mathbf{x}_i^T \boldsymbol{\beta}$$

3. Binary Outcomes

$$Y_i \in \{0, 1\}, \quad Y_i \sim \text{Bernoulli}(p_i)$$


Linearly Separable (Assumption Met)
Feature 1
Feature 2
Class 0
Class 1
Linear boundary


Non-Linearly Separable (Assumption Violated)
Feature 1
Feature 2
Class 0
Class 1
Linear boundary (fails)

Deriving a Loss Function:

Step 1 - Maximum Likelihood Estimation

Bernoulli PMF (Probability Mass Function):

$$P(Y_i = y_i \mid \mathbf{x}_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$$

$$y_i = 1: \quad p_i^1(1 - p_i)^0 = p_i$$

$$y_i = 0: \quad p_i^0(1 - p_i)^1 = 1 - p_i$$

Joint Probability of entire dataset:

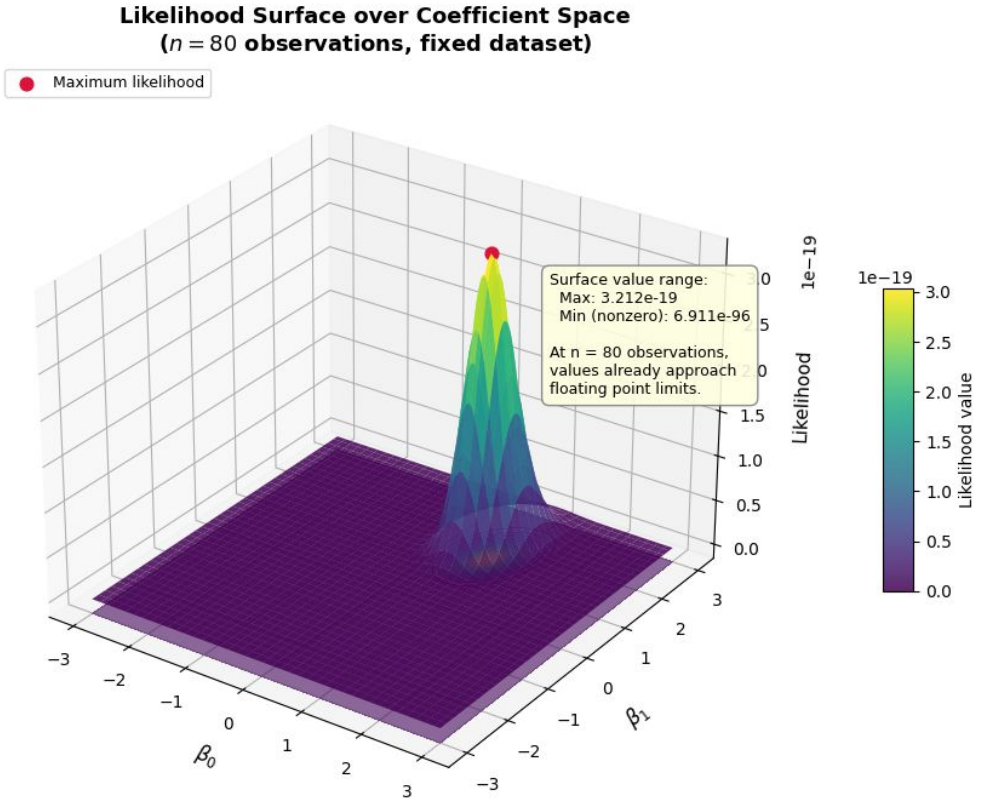
$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$$

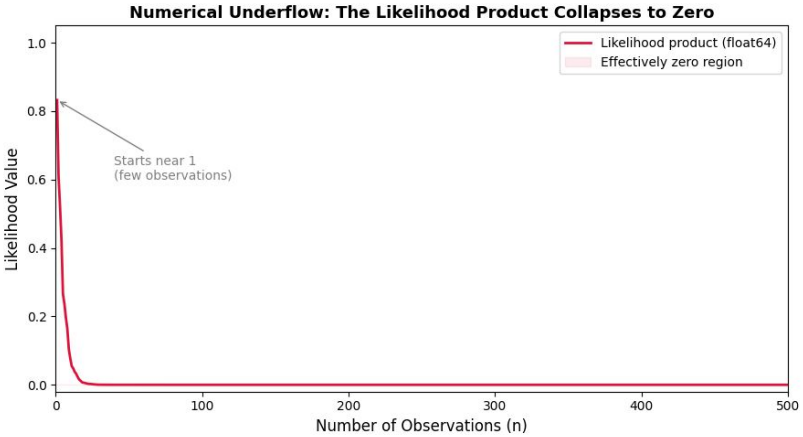
Goal:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta})$$

(This product form is mathematically inconvenient. Next, we will take the log.)

Maximizing the Likelihood is Computationally Inefficient





$$\mathcal{L}(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i}$$

$$\hat{\beta} = \arg\max_{\beta} \mathcal{L}(\beta)$$

Deriving a Loss Function:

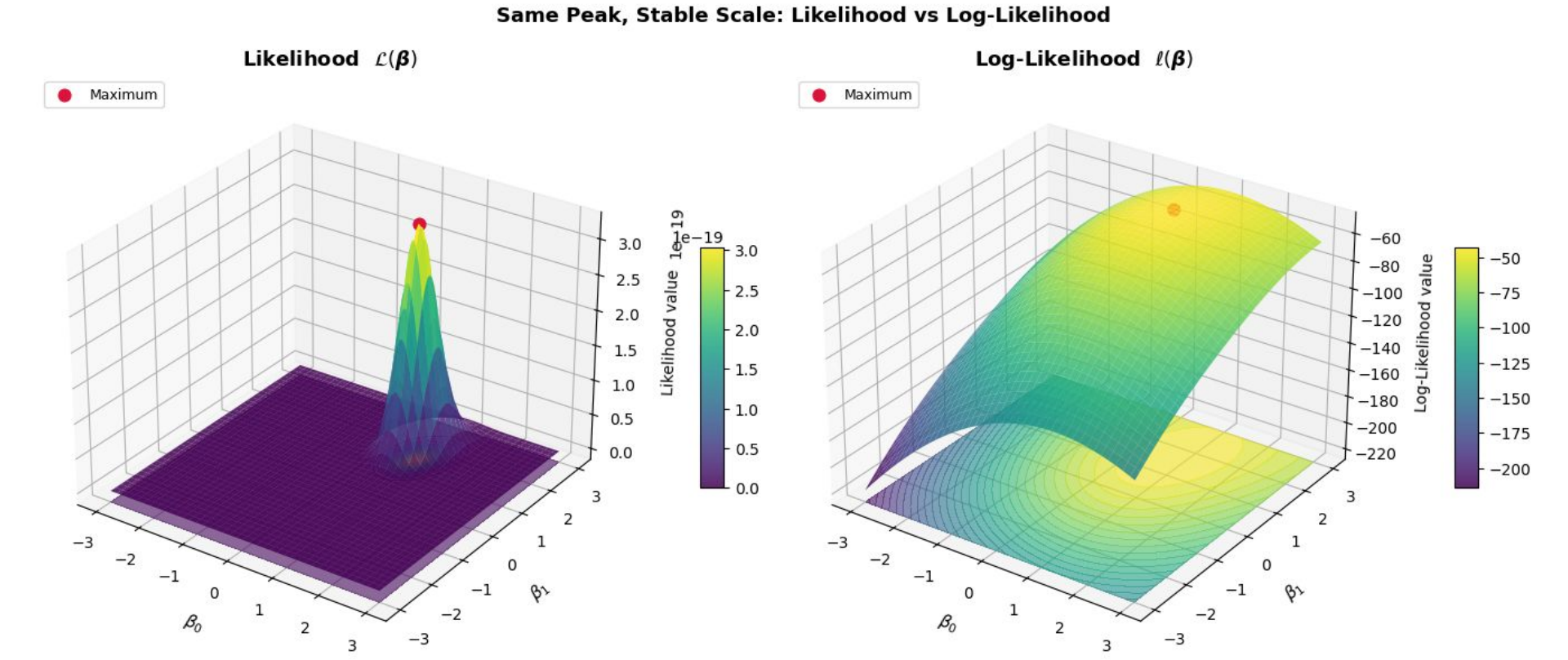
Step 2 - Log Likelihood

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \log \mathcal{L}(\boldsymbol{\beta})$$

$\log$ is strictly monotonically increasing $\implies$ $\arg\max$ is preserved

$$\ell(\boldsymbol{\beta}) = \log \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i} = \sum_{i=1}^{n} \log \left[p_i^{y_i} (1 - p_i)^{1 - y_i} \right]$$

$$= \sum_{i=1}^{n} \left[y_i \log p_i + (1 - y_i) \log (1 - p_i) \right]$$


Same Peak, Stable Scale: Likelihood vs Log-Likelihood
Likelihood $\mathcal{L}(\boldsymbol{\beta})$
Maximum
1e−19
Likelihood value
β_0
β_1
Log-Likelihood $\ell(\boldsymbol{\beta})$
Maximum
Log-Likelihood value
β_0
β_1

Deriving a Loss Function:

Step 3 - From Log-Likelihood to a Loss Function

Negate the log-likelihood and reformulate to find a minimum:

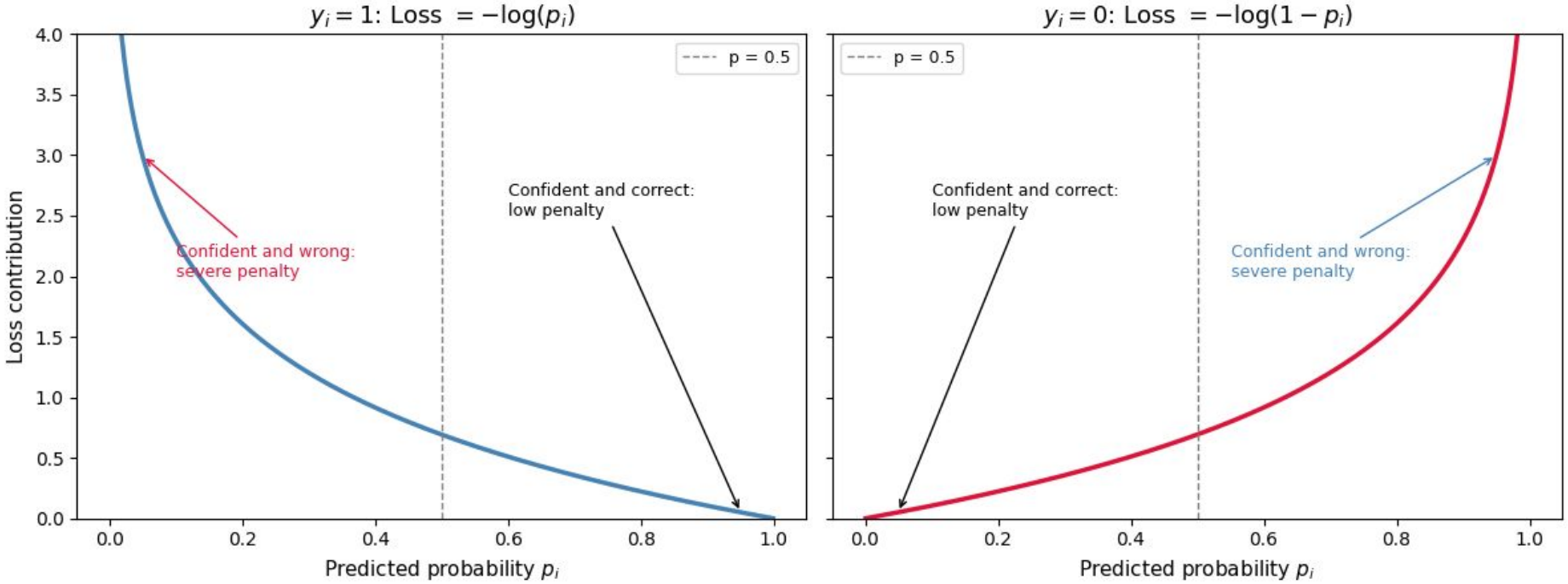
$$\arg\max_{\beta} \ell(\beta) = \arg\min_{\beta} \; -\ell(\beta)$$

Divide by n to ensure minimization is not affected by dataset size (**Binary Cross-Entropy Loss**):

$$\mathcal{J}(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]$$

Or simply:

$$\mathcal{J}(\beta) = \begin{cases} -\log p_i & \text{if } y_i = 1 \\ -\log(1-p_i) & \text{if } y_i = 0 \end{cases}$$



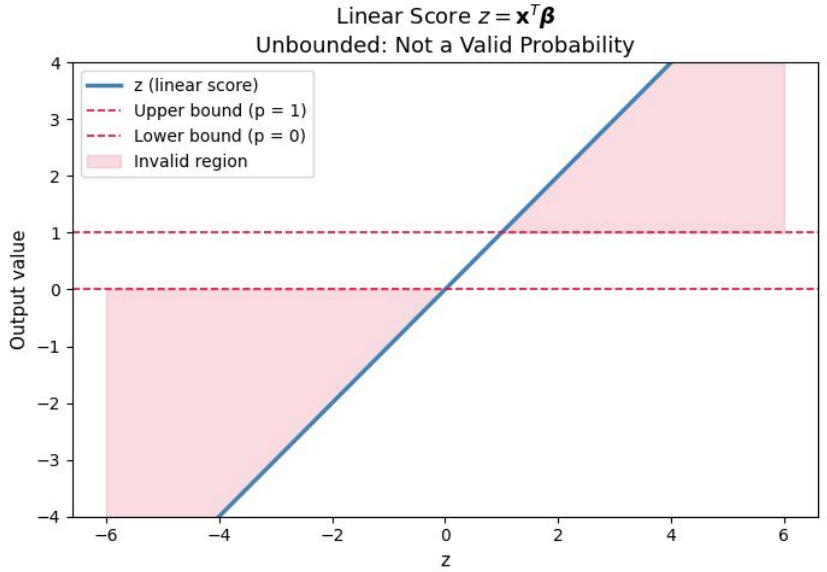
$$\mathcal{J}(\boldsymbol{\beta}) = \begin{cases} -\log p_i & \text{if } y_i = 1 \\ -\log(1 - p_i) & \text{if } y_i = 0 \end{cases}$$

We have a loss function. But p still has no definition. Where does the probability actually come from?

The Linear Score is Not Enough

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k = \mathbf{x}^T \boldsymbol{\beta}$$

$z \in (-\infty, +\infty)$ but we need $p \in (0, 1)$



We need $f(z)$ such that:

(1) $0 < f(z) < 1 \quad \forall z \in \mathbb{R}$

(2) $f'(z) > 0$ (monotonically increasing)

(3) $f'(z)$ exists everywhere (differentiable)

Introducing: The Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Derived from the log-odds assumption:

$$\log \frac{p}{1-p} = z \implies p = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}$$

Satisfies all requirements:

$$(1) \quad \sigma(z) \in (0, 1) \quad \forall z \in \mathbb{R}$$

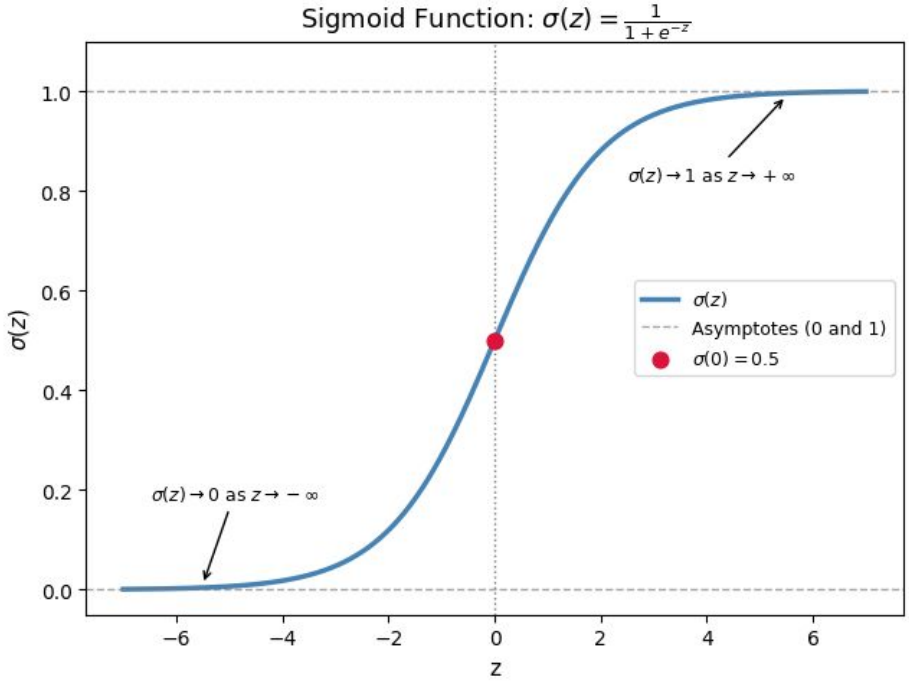
$$(2) \quad \sigma'(z) > 0 \quad \forall z \in \mathbb{R}$$

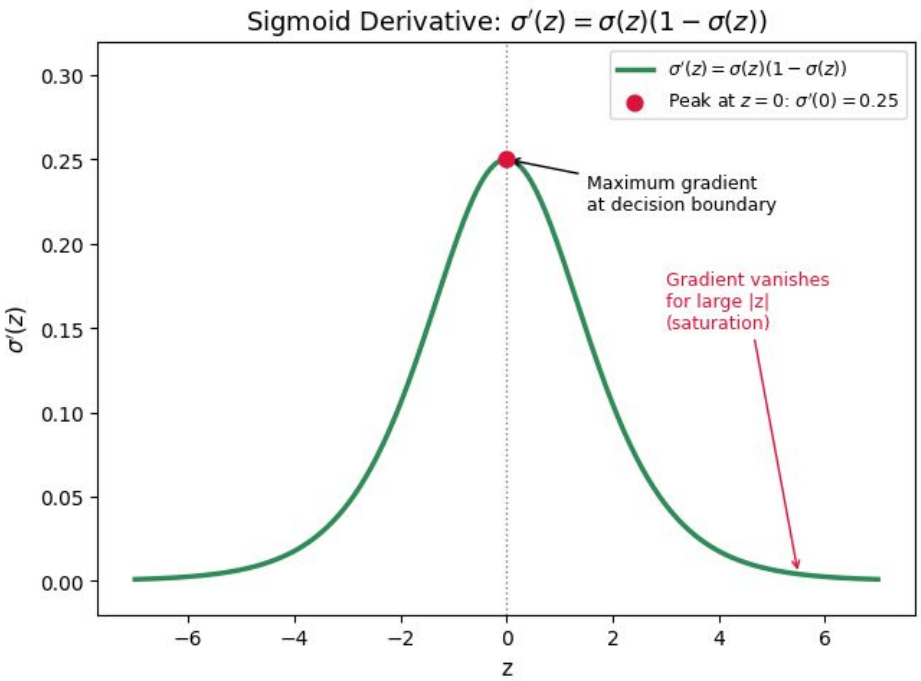
$$(3) \quad \sigma'(z) = \sigma(z)\,(1 - \sigma(z))$$

Derivative of the Sigmoid:

$$\sigma'(z) = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}}$$

$$= \sigma(z)\,(1 - \sigma(z))$$


Sigmoid Function: $\sigma(z) = \frac{1}{1+e^{-z}}$
$\sigma(z) \to 1$ as $z \to +\infty$
$\sigma(z) \to 0$ as $z \to -\infty$
$\sigma(z)$
Asymptotes (0 and 1)
$\sigma(0) = 0.5$
$\sigma(z)$
z


Sigmoid Derivative: $\sigma'(z) = \sigma(z)(1-\sigma(z))$
$\sigma'(z) = \sigma(z)(1-\sigma(z))$
Peak at $z = 0$: $\sigma'(0) = 0.25$
Maximum gradient
at decision boundary
Gradient vanishes
for large |z|
(saturation)
$\sigma'(z)$
z

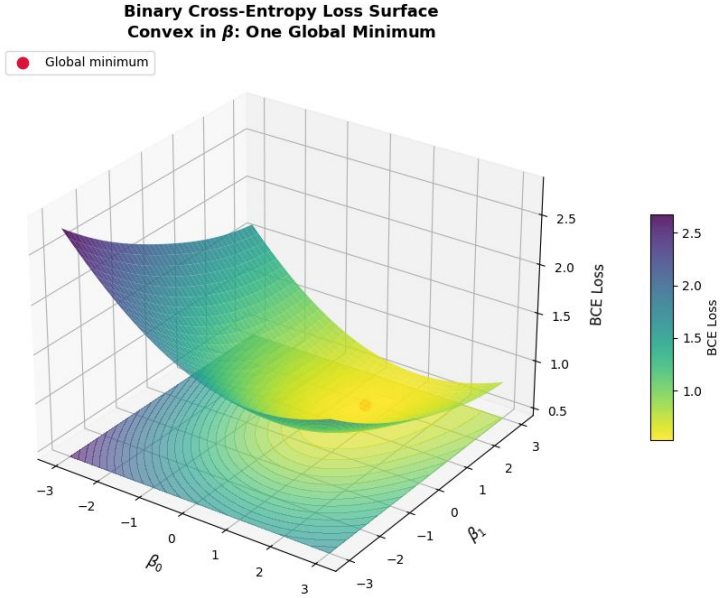
Putting It All Together: BCE & the Sigmoid

$$\mathcal{J}(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]$$

$$p_i = \sigma(z_i) = \frac{1}{1+e^{-z_i}}, \quad z_i = \mathbf{x}_i^T\boldsymbol{\beta}$$

$$\mathcal{J}(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \sigma(\mathbf{x}_i^T\boldsymbol{\beta}) + (1-y_i)\log\big(1-\sigma(\mathbf{x}_i^T\boldsymbol{\beta})\big)\right]$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathcal{J}(\boldsymbol{\beta})$$



$\mathcal{J}(\boldsymbol{\beta})$ is convex in $\boldsymbol{\beta} \implies$ gradient descent finds the global minimum

Finding the Minimum: Gradient Descent

- **Gradient descent** is an iterative optimization algorithm.
- The gradient tells us the direction of steepest ascent. **We move in the opposite direction to descend**.
- The size of the step is controlled by the **learning rate**, a hyperparameter that balances convergence speed against stability.
- Gradient descent iterates until the loss stops decreasing meaningfully, or a maximum number of iterations is reached.

Gradient Descent: Update Rule Formulation

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \alpha \nabla_{\boldsymbol{\beta}} \mathcal{J}(\boldsymbol{\beta}^{(t)})$$

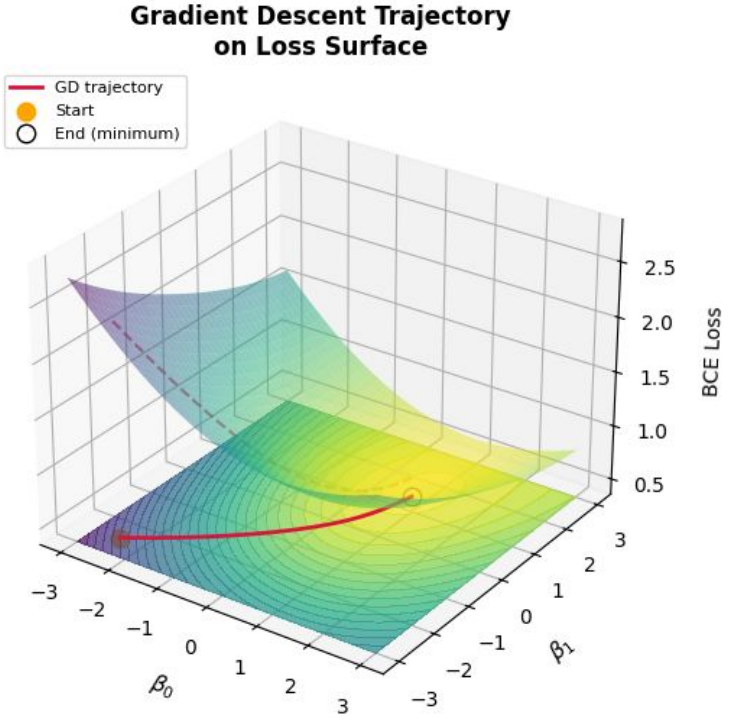
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} = \frac{\partial \mathcal{J}}{\partial p_i} \cdot \frac{\partial p_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial \boldsymbol{\beta}}$$

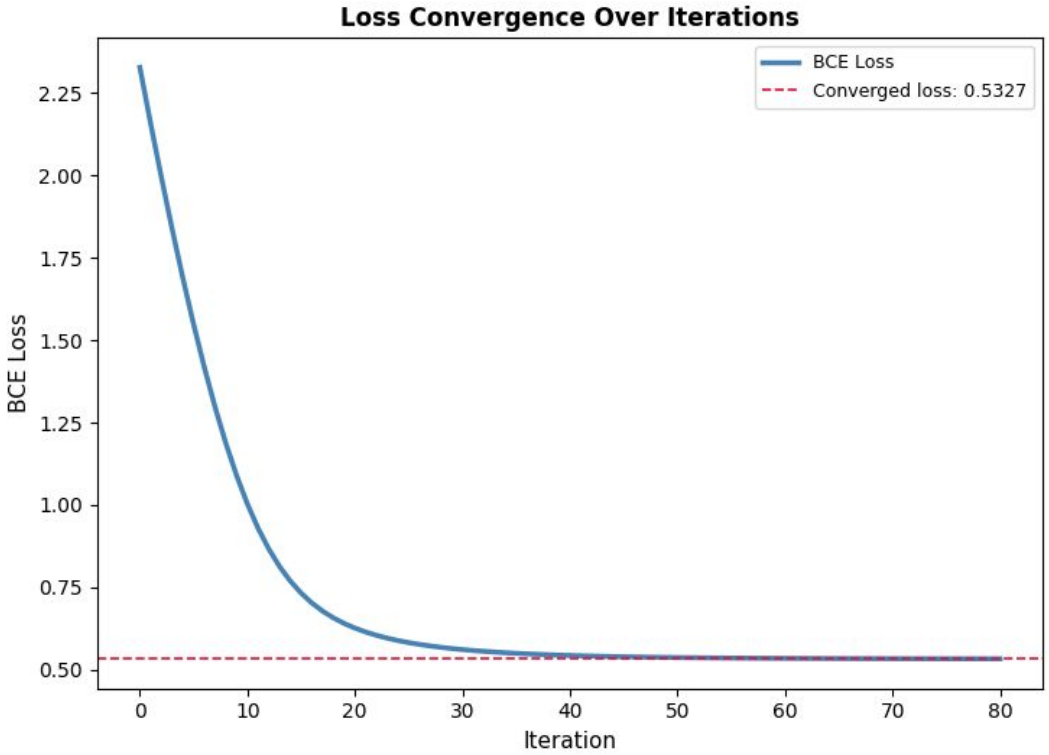
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} = -\frac{1}{n} \sum_{i=1}^{n} \left[\frac{y_i}{\sigma_i} - \frac{1 - y_i}{1 - \sigma_i} \right] \sigma_i (1 - \sigma_i) \mathbf{x}_i$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} = \frac{1}{n} \sum_{i=1}^{n} (\sigma_i - y_i) \mathbf{x}_i$$

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \frac{\alpha}{n} \sum_{i=1}^{n} \left(\sigma(\mathbf{x}_i^T \boldsymbol{\beta}^{(t)}) - y_i \right) \mathbf{x}_i$$

Gradient Descent on the BCE Loss Surface





$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \frac{\alpha}{n} \sum_{i=1}^{n} \left(\sigma(\mathbf{x}_i^T \boldsymbol{\beta}^{(t)}) - y_i \right) \mathbf{x}_i$$

Thank you!