Binary class logistic regression with a regularization term, minimizes the following custom cost function

$$\mathscr{L}(\boldsymbol{w}) = \sum_{i=1}^{N} [-y_i \log(\mu_i) - (1-y_i)\log(1-\mu_i)] + \lambda w_1^2.$$

Here, $\mu_i = \text{sigm}(w_0 + w_1 x_{1,i} + w_2 x_{2,i})$, where sigm$(\cdot)$ is the sigmoid function and λ is the regularization parameter. Suppose λ is a very large number, i.e., $\lambda \to \infty$. Here, the variable x_1 is plotted on the horizontal (x-axis) and the variable x_2 is plotted on the vertical (y-axis). What can you say about decision boundary? [10 marks]

- (●) a. Decision boundary becomes horizontal line, and y axis crossing is determined by w_0.
- ( ) b. Decision boundary becomes horizontal line and passing trough origin.
- ( ) c. None of the answers provided are correct
- ( ) d. Decision boundary becomes horizontal line, and y axis crossing is determined by w_1.

The decision boundary in logistic regression is determined by the equation:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

When regularization (L2) is applied, and $\lambda \to \infty$, the regularization forces the weights w_1 and w_2 to approach 0. As w_1 and w_2 become very small due to heavy regularization, the equation simplifies to:

$$w_0 = 0$$

This results in a decision boundary where x_2 (the vertical axis) becomes irrelevant, and the boundary is a horizontal line determined by the intercept w_0. Therefore, the crossing point on the y-axis is determined by w_0.

- **a. Decision boundary becomes a horizontal line, and y-axis crossing is determined by w_0.**

Which of the following statements is true regarding the performance of the saga solver in Logistic Regression? [10marks]

- a. The SAGA solver performs well even when features are on different scales.
- b. The SAGA solver is unaffected by the scale of the features and automatically normalizes them.
- c. The SAGA solver performs poorly when features are on different scales and may require feature scaling for optimal performance.
- d. The SAGA solver can only be used with binary classification problem

c. The SAGA solver performs poorly when features are on different scales and may require feature scaling for optimal performance.

While the SAGA solver can handle different types of data, its performance can be improved significantly by standardizing or normalizing the feature scales.

Feature	SAGA Solver	LIBLINEAR Solver
Type	Stochastic gradient descent	Coordinate descent
Scalability	Efficient for large datasets	Efficient for large datasets
Regularization	Supports L1 and L2	Supports L1 and L2
Feature Scaling	May require feature scaling	Generally requires feature scaling
Convergence	Good convergence for large data	Fast convergence, especially for smaller datasets
Use Cases	Works well for sparse data	Best for linear models
Multi-class Support	Supports multi-class with softmax	Supports multi-class
Implementation	Part of scikit-learn	Part of scikit-learn
Speed	Faster for very large datasets	Fast for smaller datasets
Memory Usage	May use more memory with large datasets	Generally lower memory usage

Suppose you have a categorical feature with the categories 'red', 'blue', 'green', 'blue', 'green'. After encoding this feature using label encoding, you then apply a feature scaling method such as Standard Scaling or Min-Max Scaling. Is this approach correct? [10marks]

- a. Yes, scaling is necessary to ensure all features have the same range
- b. No, scaling after label encoding is not appropriate because it creates a misleading order among categories.
- c. None of the answers provided are correct
- d. Yes, label encoding followed by scaling is appropriate for categorical features

b. No, scaling after label encoding is not appropriate because it creates a misleading order among categories.

Label encoding assigns integer values to categories, which can imply a false ordinal relationship. Scaling these encoded values can further misrepresent the data. Categorical features are typically better handled through one-hot encoding rather than label encoding followed by scaling.

$$\text{standard scaling} = \frac{x - \mu}{\sigma}$$

$$\text{min-max scaling} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Why is one-hot encoding generally preferred over label encoding for categorical features in machine learning? [10marks]

- [x] a. One-hot encoding represents each category as a binary vector (e.g., red = [1, 0, 0], blue = [0, 1, 0], green = [0, 0, 1]), preventing any implied ranking.
- [ ] b. One-hot encoding requires fewer columns than label encoding, making the dataset smaller.
- [ ] c. One-hot encoding reduces the dimensionality of the data compared to label encoding.
- [ ] d. One-hot encoding creates a numerical order among categories, making it easier for algorithms to interpret.

Clear my choice

One-hot encoding prevents the algorithm from assuming any ordinal relationship between categories, which is a limitation of label encoding where categories are assigned arbitrary integer values.

Feature	One-Hot Encoding	Label Encoding
Representation	Each category is represented as a binary vector.	Each category is represented as a unique integer.
Dimensionality	Increases dimensionality (one column per category).	Reduces dimensionality (one column for all categories).
Implied Order	No implied order among categories (avoids ordinal relationships).	Implies an ordinal relationship (e.g., category 1 < category 2).
Suitability	Best for non-ordinal categorical data.	Suitable for ordinal categorical data.
Sparsity	Results in sparse matrices (many zeros).	Results in dense matrices.
Memory Usage	Can consume more memory with many categories.	More memory efficient with fewer columns.
Use Cases	Commonly used in machine learning models (e.g., neural networks).	Often used with tree-based algorithms (e.g., decision trees).

Consider following loss function for binary class logistic regression.

$$\text{Binary Cross-Entropy (BCE)} = \frac{1}{N}\sum_{i=1}^{N}\left[-y_i \log(\mu_i + \epsilon) - (1-y_i)\log(1-\mu_i + \epsilon)\right].$$

Here, $\mu_i = \text{sigm}(\boldsymbol{w}^T \boldsymbol{x}_i)$ with features $\boldsymbol{x}_i$, and sigm$(\cdot)$ is the sigmoid function. Further, ϵ is small positive number. What is the usage of ϵ. [10marks]

- (●) a. It helps to maintain numerical stability
- ( ) b. To act as a regularization term in the loss function
- ( ) c. None of the answers provided are correct
- ( ) d. It ensures that all predictions are positive

a. It helps to maintain numerical stability.

The small positive number ϵ is added to prevent issues with taking the logarithm of zero, which can occur when predictions are exactly 0 or 1. This helps maintain numerical stability during computations.

"logreg = LogisticRegression(solver='saga')" initializes a logistic regression model with the SAGA solver. What is the usage of solver ? [10marks]

- a. To determine the number of features to include in the model.
- b. To specify the method used for data preprocessing before fitting the model.
- c. To set the maximum number of iterations for training the model.
- d. Solver is used to minimize the loss function of the logistic regression during model training.

Od. Solver is used to minimize the loss function of the logistic regression during model training.

The "solver" in logistic regression specifies the optimization algorithm used to minimize the loss function, which is crucial for fitting the model to the training data. Different solvers can have different properties, such as speed and convergence behavior.

Suppose that variables x_1 representing the number of practice hours per week, x_2 representing the student's previous musical experience (on a scale from 1 to 10), and $y = 1$ indicating student passing the final performance. After training a logistic regression model $p(y = 1|\boldsymbol{x}, \boldsymbol{w}) = f(w_0 + w_1 x_1 + w_2 x_2^2)$, where $f(\cdot)$ is a function, that maps $w_0 + w_1 x_1 + w_2 x_2^2$ to a probability value. We obtained the following estimated coefficients: $w_0 = -5$, $w_1 = 0.2$, and $w_2 = 0.1$. Function f is defined as

$$f(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } 0 \le z \le 1 \\ 1 & \text{if } z > 1 \end{cases}$$

- What is the estimated probability that a student, who practices for 15 hours per week and has a musical experience level of 6, will pass the final performance? [15marks]
- To achieve a 90% chance of passing the final performance, how many hours of practice does the student need if their experience level is 6? [15marks]

a. Estimated probability of passing is 0 and To achieve a 90.0% chance of passing the student needs approximately 26.5 hours of practice.

b. None of the answers provided are correct

c. Estimated probability of passing is 0.1978 and To achieve a 90.0% chance of passing the student needs approximately 32.99 hours of practice.

d. Estimated probability of passing is 1 and To achieve a 90.0% chance of passing the student needs approximately 11.50 hours of practice.

$$P(y=1 \mid x, w) = f(w_0 + w_1 x_1 + w_2 x_2^2)$$
$$= f(-5 + 0.2 x_1 + 0.1 x_2^2)$$
$$= f(-5 + 0.2 \times 15 + 0.1 \times 6^2)$$
$$= f(-5 + 3 + 3.6)$$
$$= f(1.6)$$
$$= 1$$

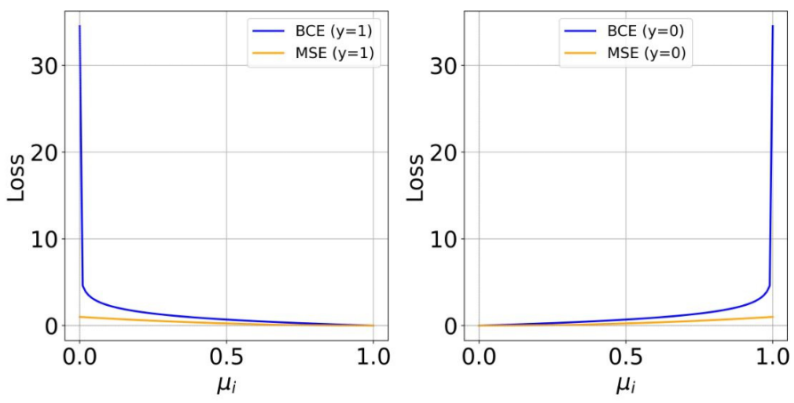
$$0.9 = -5 + 0.2 x_1 + 0.1 \times 6^2$$
$$2.3 = 0.2 x_1$$
$$x_1 = 11.5 \text{ hrs}$$

Consider following two loss functions for binary class logistic regression. [10marks]

$$\text{Binary Cross-Entropy (BCE)} = \frac{1}{N}\sum_{i=1}^{N}\left[-y_i \log(\mu_i+\epsilon) - (1-y_i)\log(1-\mu_i+\epsilon)\right].$$

$$\text{Mean Squared Error (MSE)} = \frac{1}{N}\sum_{i=1}^{N}(y_i-\mu_i)^2.$$

Here, $\mu_i = \text{sigm}(\boldsymbol{w}^T\boldsymbol{x}_i)$ with features $\boldsymbol{x}_i$, and sigm$(\cdot)$ is the sigmoid function. Further, ϵ is small positive number. Figure shows change of loss function with respect to predicted class μ_i (left: for class label = 1 and right: class label =0).



Which of the following is true ?

- a. Binary Cross-Entropy (BCE) penalize incorrect predictions more severely than MSE does, which helps in better convergence for classification tasks.
- b. MSE provides deterministic outputs, while BCE provides probabilistic outputs
- c. BCE provides deterministic outputs, while MSE provides probabilistic outputs
- d. None of the answers provided are correct

Left Graph (Label = 1, $y_i = 1$):

- **BCE (blue line)**: The loss decreases sharply as the predicted probability μ_i approaches 1. If μ_i is close to 0 (the prediction is very wrong), the loss is very large.
- **MSE (yellow line)**: The loss is relatively low compared to BCE, even for incorrect predictions (i.e., when μ_i is far from 1). The decrease is more gradual as μ_i approaches 1.

Right Graph (Label = 0, $y_i = 0$):

- **BCE (blue line)**: Similar behavior as in the left graph, but the loss is very large when μ_i is near 1 (incorrect prediction), and decreases sharply as μ_i approaches 0 (correct prediction).
- **MSE (yellow line)**: Again, the loss is more gradual compared to BCE, and increases as μ_i deviates from 0, but not as severely as BCE.

Key Takeaways from Graphs:

- **BCE** has more extreme penalties for incorrect predictions than **MSE**, especially when the predicted probability μ_i is far from the actual label.
- **MSE** has a more gradual loss curve, meaning it doesn't penalize incorrect predictions as severely, which is why BCE is preferred for classification tasks while MSE is more suited for regression.

↓

Binary Cross-Entropy (BCE) penalizes incorrect predictions more severely than **Mean Squared Error (MSE)**, particularly in classification tasks. This makes BCE more suitable for binary classification problems, where it helps the model converge better by emphasizing incorrect predictions more sharply.

This is because BCE is specifically designed for probabilistic outputs (like those in logistic regression or neural networks with sigmoid activations), while MSE is more commonly used in regression tasks where the outputs are continuous.