Physical Education, Health and Social Sciences

https://journal-of-social-education.org

E-ISSN: 2958-5996

P-ISSN: 2958-5988

Mathematical Analysis of Loss Functions in Classification and Regression

Sohail Ahmed Memon[*1], Imtiaz Ahmed[2], Shoaibullah[3]

[*1,2,3]Department of Mathematics, Shah Abdul Latif University, Khairpur Mirs.
[*1]suhail.memon@salu.edu.pk, [2]sharimtiaz2014@gmail.com, [3]shoaibpk00@gmail.com.

DOI: https://doi.org/10.63163/jpehss.v3i2.497

Abstract
Classification and Regression are commonly used techniques in machine learning. The Loss functions are foundational to machine learning, serving as quantifiable measures of prediction error though which model optimization is lead. This study presents a comprehensive mathematical analysis of prominent loss functions utilized across classification and regression tasks. We present formal definitions and investigate inherent mathematical properties such as convexity and differentiability. Their implications for the learning process, specifically within gradient-based optimization frameworks. The analysis begins with fundamental regression losses such as Mean Squared Error and Mean Absolute Error. The analysis further extends to robust alternatives such as Huber and Quantile Loss. For classification, we explore binary and categorical Cross-Entropy, Hinge Loss, Exponential Loss by clarifying their characteristics. Also, we discuss practical considerations in selecting appropriate loss functions, importance of regularization, custom loss functions design, and the critical distinction between optimization objectives and evaluations metrics. The work aims to create understanding for readers with deeper theoretical understanding of how different loss functions shape model behaviour and performance in diverse machine learning applications.

Keywords: Loss Functions; Binary Classification; Regression Analysis; Machine Learning Theory; Model Robustness

1. Introduction

The loss function is thought to be the fundamental concept for every machine learning algorithm. A loss function quantifies the discrepancy between a predicted output of model and the true or observed value for a given data point. It provides a numerical measure of how “wrong” a model’s prediction is. It serves as the guiding signal for the learning process and therefore this single value is critical. During model training, an optimization algorithm, most commonly gradient descent or its variants, iteratively adjusts the internal parameters of model with explicit goal of minimizing this loss. The final aim is to find a set of parameters that results in the lowest possible average loss across the entire training dataset. Since the ‘loss’ term generally refers to the error on a single data instance, the aggregated sum or average of these individual losses across the whole dataset is termed the ‘cost function’ or ‘objective function’ [1]. The field of machine learning vastly categorizes tasks into two primary types, one is regression and the other is classification. Regression problems involve prediction of a continuous numerical output, such as house prices, temperature, or stock prices [2]. While classification problems aim to predict discrete categorical labels, like whether an email is spam or not, identifying the type of animal in an image, or predicting a patient’s disease diagnosis.

The inherent nature of these tasks dictates the choice of loss function. For continuous predictions in regression, errors are typically measured by the numerical difference between predicted and true values. In classification problems, the scenario is different, a simple numerical difference is often meaningless. Instead, the more focus is about the likelihood of correct category assignment and the confidence in classifications. This fundamental difference necessitates distinct families of loss functions tailored to each problem type, each designed to capture the specific nuances of prediction error in its respective domain [3]. Mathematical analysis is essential to understand loss functions for their implementations in machine learning models. To treat loss function as an error is not sufficient. A comprehensive mathematical based analysis is essential for many reasons. Whether loss functions are convex or ensure differentiability at every point. If these functions lack convexity or are not differentiable at any point then what could be the alternatives. All such discussion is a top priority. Convexity ensures the convergence to a global minimum for a gradient-based optimization. It avoids suboptimal local minima. Differentiability property of loss function ensures that gradients can be reliably computed. It is responsible for dictating the direction and magnitude of updated parameters [4]. It is essential to understand the deeply rooted principles of loss functions embedded in Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MA) for data distribution and noise[5]. Such analysis of fundamentals yields proper information about their behaviour. Although, the mathematical construction of loss functions directly affects the robustness of model to outliers. Sensitivity to noisy data and even the interpretability of learned parameters of loss functions can be understood by such analysis [2], [6]. Our study reveals why certain loss functions might be preferred over others in specific situations, even if these initially appear to measure the same concept of 'error'. This analysis work is presented in different sections of this article. Section 2 begins with fundamental definitions. We have covered common loss functions used for the regression problems in section 3. The log functions used for binary classification problems are discussed in section 4. The overall analysis as conclusion is given in section 5.

2. Fundamentals of Loss Functions and Optimizations

This section explores the understanding of loss functions which begins with their precise mathematical definitions and the context in which they operate within machine learning paradigms. Here, we present a groundwork by formalizing what a loss function is, its role in empirical risk minimization, and the crucial mathematical properties that dictate its utility in optimization. A loss function, denoted as $L(y, \hat{y})$, is a mathematical function that measures the penalty for a model's incorrect or imprecise prediction. Here y represents the true (observed value) and $\hat{y}$ represents the value predicted by the machine learning model. The output of $L(y, \hat{y})$ is a non-negative real number, where a value of zero typically indicates a perfect prediction. It means that the predicted values $\hat{y}$ becomes true value y. Whereas, the higher values indicate greater discrepancies [7]. Although, a loss function quantifies error for a single data point, the overall performance of a model across its training data is examined by an objective function or cost function which is usually denoted as $J(\theta)$. This function is the average or sum of individual losses over all n data points in the dataset [1]. If θ represents the set of all parameters within a model, and $\hat{y}_i(\theta)$ is the prediction for the i-th data point y_i given these parameters, the cost function is commonly expressed as [3]:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L\big(y, \hat{y}(\theta)\big) \qquad (1)$$

The main objective of training a machine learning model is to find the best set of parameters θ^* that minimizes this cost function over the training data:

$$\theta^* = \arg\min_{\theta} J(\theta) \quad (2)$$

The notion of minimizing the cost function is formally rooted in Empirical Risk Minimization (ERM) [8]. In ERM, the true underlying risk is approximated to minimize the empirical risk, which is an approximation of loss on the expected loss on unseen data. The ERM is basically the average loss computed over the available training data. The main concept is that the model will generalize effectively to new or unseen data provided that it has performed well on training data. For such performance, it is assumed that the training data is representative of the underlying data distribution. The selection of a loss function is therefore a direct commitment to a specific form of empirical risk, guiding the model to learn patterns that minimize this selected measure of error [8]. The mathematical properties are considered crucial for a loss function to be effectively minimized by gradient-based optimization algorithms, these properties are as follows:

- **Convexity**: A function $f(x)$ is convex if, for any two points x_1 and x_2 in its domain and any $\lambda \in [0,1]$, the following holds:

  $$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \quad (3)$$

  Intuitively, the graph of a convex function lies below the line segment connecting any two points on the graph. For optimization, convexity is highly desirable because it ensures that any local minimum found by a gradient-based technique is also a global minimum [9].
- **Differentiability**: The loss function must be differentiable for a gradient-based optimization method to work. Accordingly, its gradient can be computed at every point. The gradient $\nabla J(\theta)$ indicates the direction of the steepest ascent of cost function. The cost is iteratively reduced by movement of gradient in the opposite direction [10]. Various loss functions fail to be differentiable at every point where they might have sharp corners. In such cases, the concept of subgradient is used and this approach allows optimization algorithms to still make progress [4]. Although, implementing subgradient approach is complicated to handle but many practical algorithms can effectively use it [11].

The optimization algorithm is based on Gradient Descent, which is considered a cornerstone. The Gradient Descent is driven directly by the gradient of loss function. It is an iterative first-order optimization algorithm used to find the minimum of a function [12]. It starts with an initial set of model parameters $\theta^{(k)}$, the algorithm updates these parameters by taking a step proportional to the negative of the gradient of the cost function $J(\theta)$ with respect to θ.

The update rule for Gradient Descent is given by:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla J\left(\theta^{(k)}\right) \quad (4)$$

In equation (4):

- $\theta^{(k+1)}$ represent the updated model parameters.
- $\theta^{(k)}$ represent the current model parameters.
- α is the learning rate. Although, it is a small scalar that determines the size of the step taken in the direction of the negative gradient but is carefully chosen to get efficient convergence. The algorithm might overshoot the minimum, if it is too large. With too small choice, the algorithm might take longer time to converge.
- $\nabla J\left(\theta^{(k)}\right)$ is the gradient of the cost function with respect to parameters θ, evaluated at $\theta^{(k)}$. It points in the direction of the steepest increase of the cost function. Its subtraction leads in the direction of steepest decrease, aiming towards the minimum.

The gradient descent, through its iterative nature, enables machine learning models to learn from data by progressively refining their parameters to minimize prediction errors.

3. Loss Functions for Regression

Generally, the regression tasks predict the continuous numerical values where loss functions are supposed to appropriately penalize the magnitude of error. The loss functions that are used in regression tasks are explained in this section. We explain these functions with their mathematical properties and how they become effective for the regression task.

3.1 Mean Squared Error (MSE)

The Mean Squared Error, also referred to as L2 Loss, is frequently used as a loss function regression problems. It quantifies the average of the squares of the errors. An error can be defined as the difference between the actual value and the predicted value.

For a single data point, the squared error loss is given by:

$$L(y, \hat{y}) = (y - \hat{y})^2 \tag{5}$$

where y is the true value and $\hat{y}$ is predicted value.

When it is applied across a dataset of Nsamples, the MSE is the average of these individual squared errors. Thus, it becomes the cost function given as follows [4] :

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i - \hat{y}_i(\theta)\big)^2 \tag{6}$$

where $\hat{y}(\theta)$ is the predicted value of model for the i-th true value y_i which depends on the parameter θ of the model.

- **Mathematical Properties:**

  Convexity: The MSE is convex, it is considered highly desirable property for the optimization.

  Differentiability: It is continuously differential at every point. It is considered a suitable candidate for gradient-based optimization algorithms. Using MSE, the gradient is always easy to compute.

  Quadratic Nature: The squared terms ensure that large errors are penalized disproportionately more than small errors.
- **Geometric Interpretation:** Minimizing MSE corresponds to finding the line or curve that minimizes the Euclidean distance (L2 norm) between the observed values and the predicted values. It strongly quantifies the average squared deviation between actual values and predicted values.
- **Gradient:** For a single loss function $L(y, \hat{y})$=$(y - \hat{y})^2$, the partial derivative with respect to predicted $\hat{y}$ is:

  $$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}(y - \hat{y})^2 = -2(y - \hat{y}) \tag{7}$$

  The negative sign indicates that if $\hat{y}$ is too low, the gradient is positive, pushing $\hat{y}$ higher to reduce the loss, and vice-versa. Thus, linear gradient makes the optimization straightforward.
- **Impact on Model:** MSE is highly sensitive to outlier due to its quadratic nature. This can lead to models that perform poorly on the majority of data if outliers are present [13]. MSE contributes to Gaussian Noise when errors are assumed to be normally distributed and homoscedastic (constant variance) [10].

3.2 Mean Absolute Error (MAE)

The Mean Absolute Error which is also known as L1 Loss which measures the average of the absolute differences between predictions and actual observations. Whereas, MSE makes a linear penalty for errors [14].

For a single data point, the absolute error loss is:

$$L(y,\hat{y}) = |y - \hat{y}| \tag{8}$$

The cost function based on equation (8) is:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i(\theta)| \tag{9}$$

- **Mathematical Properties:**

  Convexity: MAE is also a convex function, ensuring that a global minimum can be found.
- **Differentiability:** The main difference between MSE and MAE is that MAE is not differential at $y = \hat{y}$. At this point, the absolute value function has a sharp corner which means the derivative is undefined.
- **Linear Penalty:** The penalty for an error is indicated to grow linearly with the magnitude of the error.
- **Geometric Interpretation:** When MAE minimizes, it corresponds to finding the line or curve that minimizes the Manhattan distance (L1 norm) between true values and predicted values.
- **Gradient (Subgradient):** The derivative of $|x|$ is 1 for $x > 0$, -1 for $x < 0$, and undefined at $x = 0$. Thus, for $L(y,\hat{y})$=$|y - \hat{y}|$, the partial derivative with respect to $\hat{y}$ is [15]:

$$\frac{\partial L}{\partial \hat{y}} = \begin{cases} -1 & \text{if } y - \hat{y} > 0, & (\hat{y} < y) \\ 1 & \text{if } y - \hat{y} < 0, & (\hat{y} > y) \\ \text{undefined} & \text{if } y - \hat{y} = 0, & (\hat{y} = y) \end{cases} \tag{10}$$

At the point where the derivative is undefined, optimization algorithms often rely on a subgradient. The subgradient for $|x|$ at $x = 0$ is any value in $[-1,1]$. Optimization algorithms often handle this issue gracefully by using optimization techniques such as Adam or RMSprop that are less sensitive to these non-differentiable points.

- **Impact on Model**:

  Robustness to Outliers: Linear penalty by MAE makes it significantly more robust to outliers than MSE [2]. This is because large errors contribute proportionally, therefore a few extreme values do not dominate the total loss.

  Median Estimation: Unlike MSE which finds the conditional mean, minimizing MAE is equivalent to finding the conditional median of the target variable. Thus, this property contributes to its robustness.

3.3 Huber Loss (Smooth L1 Loss)

The Huber Loss which is also known as Smooth L1 Loss, introduced by Peter Huber in 1964 as a robust alternative to MSE. It was designed to be less sensitive to outliers while retaining differentiability [16].

The Huber Loss combines the advantages of squared loss and absolute loss. It is defined as:

$$L(y,\hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \le \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{if } |y - \hat{y}| > \delta \end{cases} \tag{11}$$

In equation (11), $\delta > 0$ is a threshold parameter. If error is small (less than δ), Huber Loss works as squared loss, easing smooth optimization. If error is large, it works like absolute loss, reducing the impact of outliers.

Its impact on model is that the Huber Loss leads to models that are more robust to outliers than those trained with squared loss. It surely retains differentiability and enables efficient gradient-based optimization. When the data contains noise or occasional extreme values, it is very helpful.

3.4 Quantile Loss

The Quantile is often used in problems related to quantile regression and it has applications in financial forecasting, risk assessment or asymmetric cost functions. Its goal is to estimate a significant quantile or median of the response variable. Mathematically, the Quantile Loss $L_\tau(y, \hat{y})$ for true value y, a predicted value $\hat{y}$, and desired quantile $\tau \in (0,1)$ is defined as:

$$L_\tau(y, \hat{y}) = \begin{cases} \tau(y - \hat{y}) & \text{if } \hat{y} < y \\ (1-\tau)(y - \hat{y}) & \text{if } \hat{y} \geq y \end{cases} \tag{12}$$

The utilization of Quantile Loss allows the model to predict different conditional quantiles which is helpful for modelling heteroscedasticity and prediction intervals. Thus, it impacts on model [17]. The choice of loss function, especially in regression tasks, influences not only the optimization process but also the robustness and interpretability of the model. We have discussed two important loss functions that address shortcomings of traditional losses from the use of Mean Squared Error (MSE) and Mean Absolute Error (MAE) are Huber Loss and Quantile Loss.

4. Loss Functions for Binary Classification

In binary classification, the goal is to assign inputs to one of two class, typically labelled as $y \in \{0,1\}$ or $y \in \{-1,1\}$, unlike regression tasks where target variable y is continuous. The machine learning models used for binary classification are referred as classifiers. The choice of loss function directly impacts the classifier's performance, learning behaviour, and theoretical bond. A loss function $L_\tau(y, \hat{y})$ which generally measures the discrepancy between true label y and model's predicted value $\hat{y}$, so the prediction $\hat{y}$ may be a score, a probability, or a hard label [2]. We describe here some of the commonly used loss functions for binary classification problems.

4.1 Binary Cross-Entropy (Log Loss)

The Binary Cross-Entropy (BCE) is also known as Log Loss. It is considered predominant loss function for training binary classifier that output a probability estimate. Given a dataset of input-label pairs (x_i, y_i) with $y_i \in \{0,1\}$ and a model producing a predicted probability $\hat{p}_i = P(y = 1|x_i)$, the BCE is for a single example is defined as:

$$L(y, \hat{p}) = \big(y\log(p) + (1+y)\log(1-p)\big) \tag{13}$$

The total loss is the average of $L(y, \hat{p})$ over all examples [5].

For a probabilistic interpretation, the BCE arises directly from the log-likelihood of a Bernoulli distribution. Maximizing the likelihood of the observed labels under a Bernoulli model, given as under:

$$P(y \mid x_i) = \hat{p}^y(1-\hat{p})^{1-y} \tag{14}$$

Equation (14) is equivalent to minimizing the sum of BCE terms. Such connection is a guarantee that the classifier's outputs are well-calibrated probability estimate.

4.2 Logistic Loss

The Logistic Loss is commonly used in logistic regression which is also a type of binary classification. In mathematical formulation, it can be defined as:

$$L(y,\hat{y}) = \log\big(1 + \exp(y\hat{y})\big) \qquad (15)$$

where $y \in \{-1,1\}$, and $\hat{y}$ is predicted value. Logistic loss is convex, smooth, and differentiable. It is considered as a best among the loss functions for binary classification gradient-based optimizations. Its impact is that it encourages well-calibrated probabilities and often performs well when class distributions are balanced [5].

4.3 Hinge Loss

The cost function of Hinge Loss is used in support vector machines (SVM), It is defined as:

$$L(y,\hat{y}) = \max(0,1 - y\hat{y}) \qquad (16)$$

Hinge Loss is convex but not differentiable at the hinge point. It ensures margin maximization by penalizing not only incorrect predictions but also correct ones which are too close to the decision boundary. Models trained using Hinge Loss tend to generalize smoothness to the margin principle. The models employing this cost function do not yield probability estimate and are less robust to noisy labels [18].

4.4 Exponential Loss

The Exponential Loss is used famously in boosting algorithms like AdaBoost. It provides a very aggressive penalty for misclassified points, it is given by:

$$L(y,\hat{y}) = \exp(-y\hat{y}) \qquad (17)$$

This cost function is convex and differentiable. It places exponentially increasing penalties on misclassified examples. However, such its usage can lead to overfitting, particularly when the dataset contains noisy labels or outliers. While exponential loss drive models to focus on complicated cases [19].

5. Conclusions

In this work, we have presented a demanding mathematical analysis of important loss functions implemented in regression and binary classification problems. For regression, we have examined Mean Squared Error (MSE), Mean Absolute Error (MAE), Huber Loss and Quantile Loss. Each of them is well explained for their distinct characteristics with their trade-offs between sensitivity to outliers, interpretability and robustness. MSE being optimal under Gaussian assumptions, suffers from high sensitivity to outliers. MAE yields a better alternative but lacks differentiability. Huber Loss interpolates between these two, yielding robustness with smooth optimization properties. Quantile Loss, on the other hand, enables estimation of conditional quantiles, making it suitable for modelling asymmetric distributions and uncertainty. For binary classification, we have discussed Binary Cross-Entropy (Log Loss), Logistic Loss, Hinge Loss, and Exponential Loss. These cost functions were analysed in terms of convexity, differentiability, margin behaviour, and probabilistic interpretation. Binary Cross-Entropy and Logistic Loss are probabilistically based and widely utilized for their compatibility with gradient-based optimizations and well-calibrated outputs. Hinge Loss, favourable for SVMs, ensures margin maximization and provides generalization in high-dimensional spaces. Exponential Loss as used in boosting algorithms, aggressively penalizes misclassifications but may be less robust to noisy problems. In our discussion, we have emphasized how mathematical properties of each loss function. We explained how convexity, smoothness, and differentiability affect the learning dynamics, convergence and predictive quality of machine learning models. By choosing suitable loss functions based on the task and data, the researchers or practitioners can improve both the effectiveness and interpretability of predictive models. This mathematical analysis servers as grounded reference for researchers and practitioners aiming to make informed decisions about loss function choice in regression and binary classification problems.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[3] T. Hastie, R. Tibshirani, and J. Friedman, “The elements of statistical learning.” Citeseer, 2009.

[4] S. Boyd, “Convex optimization,” *Camb. UP*, 2004.

[5] K. P. Murphy, “A probabilistic perspective,” *Text Book*, 2012.

[6] P. E. Hart, D. G. Stork, and J. Wiley, “Pattern classification,” 2001.

[7] J. T. Barron, “A general and adaptive robust loss function,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4331–4339.

[8] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[9] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, “Convexity, Classification, and Risk Bounds,” *J. Am. Stat. Assoc.*, vol. 101, no. 473, pp. 138–156, Mar. 2006, doi: 10.1198/016214505000000907.

[10] C. Robert, “*Machine Learning, a Probabilistic Perspective*,” *CHANCE*, vol. 27, no. 2, pp. 62–63, Apr. 2014, doi: 10.1080/09332480.2014.914768.

[11] S. Ruder, “An overview of gradient descent optimization algorithms,” Jun. 15, 2017, *arXiv*: arXiv:1609.04747. doi: 10.48550/arXiv.1609.04747.

[12] H. Robbins and S. Monro, “A stochastic approximation method,” *Ann. Math. Stat.*, pp. 400–407, 1951.

[13] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors),” *Ann. Stat.*, vol. 28, no. 2, pp. 337–407, 2000.

[14] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[15] N. Z. Shor, *Minimization methods for non-differentiable functions*, vol. 3. Springer Science & Business Media, 2012. Differentiable+Functions&ots=Bx5XmynxOi&sig=PONIfCNzSFpFY3CIoICT1RKDdmE

[16] P. J. Huber, “A robust version of the probability ratio test,” *Ann. Math. Stat.*, pp. 1753–1758, 1965.

[17] R. Koenker and G. Bassett Jr, “Regression quantiles,” *Econom. J. Econom. Soc.*, pp. 33–50, 1978.

[18] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[19] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.