African Institute For Mathematical Sciences
Master in Machine Intelligence

From Maximum Likelihood to Loss Functions in Machine Learning

Joshua Juste Emmanuel Yun Pei NIKIEMA

May 31, 2025

Abstract

This article explains, step by step, how well-known loss functions such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Cross-Entropy Loss are derived from the Maximum Likelihood Estimation (MLE) principle. Along the way, we clarify key concepts such as the i.i.d. assumption, log-likelihood, and why we minimize the negative log-likelihood. Our aim is to make this statistical foundation accessible to everyone learning machine learning.

1 Introduction

Most machine learning models are trained by minimizing a **loss function**. But why those specific loss functions? Why do we use squared error for regression, and cross-entropy for classification?

The answer lies in **statistics**, particularly in **Maximum Likelihood Estimation (MLE)**. This paper shows how these loss functions naturally emerge when we assume a probabilistic model for the data and apply MLE.

2 The i.i.d. Assumption

In supervised learning, we assume we observe n data points:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

We make the standard assumption that the data points are **independent and identically distributed (i.i.d.)**:

- **Independent**: knowing one data point tells us nothing about another.

- **Identically distributed**: all data come from the same distribution.

This allows us to express the likelihood of the full dataset as a product of individual likelihoods:

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{n} p(y_i \mid x_i; \theta)$$

3 Maximum Likelihood Estimation

Given a probabilistic model $p(y \mid x; \theta)$, MLE seeks to find the parameter θ that makes the observed data most probable.

Formally:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i \mid x_i; \theta)$$

Why take the log?

Working with products of probabilities can lead to numerical instability (very small numbers). The logarithm turns the product into a sum, which is easier to optimize:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \log \prod_{i=1}^{n} p(y_i \mid x_i; \theta) = \arg\max_{\theta} \sum_{i=1}^{n} \log p(y_i \mid x_i; \theta)$$

Why minimize the negative log-likelihood (NLL)?

Most machine learning frameworks minimize rather than maximize. So we rewrite:

$$\hat{\theta} = \arg\min_{\theta} -\sum_{i=1}^{n} \log p(y_i \mid x_i; \theta)$$

This function:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \log p(y_i \mid x_i; \theta)$$

is what we call the **negative log-likelihood**, or NLL. It becomes the loss function.

4 Case 1: Deriving MSE from MLE

Assumption: Gaussian distribution

We assume that for each input x_i, the target y_i is given by:

$$y_i = f(x_i; \theta) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

The conditional distribution is:

$$p(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(x_i; \theta))^2}{2\sigma^2}\right)$$

Taking the negative log-likelihood:

$$-\log p(y_i \mid x_i; \theta) = \frac{(y_i - f(x_i; \theta))^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$$

Summing over i and ignoring constants:

$$\mathcal{L}(\theta) \propto \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2$$

So minimizing NLL is equivalent to minimizing the **Mean Squared Error (MSE)**:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2$$

5 Case 2: Deriving MAE from MLE

Assumption: Laplace distribution

Now suppose the noise follows a Laplace distribution:

$$p(y_i \mid x_i; \theta) = \frac{1}{2b} \exp\left(-\frac{|y_i - f(x_i; \theta)|}{b}\right)$$

Taking the negative log-likelihood:

$$-\log p(y_i \mid x_i; \theta) = \frac{|y_i - f(x_i; \theta)|}{b} + \log(2b)$$

Ignoring constants and minimizing:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} |y_i - f(x_i; \theta)|$$

This gives the **Mean Absolute Error (MAE)** loss.

6 Case 3: Deriving Cross-Entropy from MLE

Assumption: Bernoulli distribution (binary classification)

Assume $y_i \in \{0, 1\}$, and $\hat{y}_i = f(x_i; \theta) \in (0, 1)$ represents the predicted probability that $y_i = 1$.

Then:

$$p(y_i \mid x_i; \theta) = \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}$$

The negative log-likelihood becomes:

$$-\log p(y_i \mid x_i; \theta) = -\left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\right]$$

Summing over all data points:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \left[-y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)\right]$$

This is the **binary cross-entropy loss**.

Extension: Softmax + Multinomial for multi-class classification

The same reasoning applies if we assume a softmax output and a categorical distribution over classes.

7 Conclusion

We have shown that:

- The loss function is the **negative log-likelihood**.
- The choice of distribution (Gaussian, Laplace, Bernoulli) defines the form of the loss.
- Classical loss functions like MSE, MAE, and Cross-Entropy naturally emerge from the MLE principle.

This statistical foundation helps us understand *why* we use certain loss functions and when they are appropriate.

Further Reading

- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.*
- Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction.*
- Christopher Bishop. *Pattern Recognition and Machine Learning.*
- `https://sebastianraschka.com`