Mean Squared Error (MSE) Tutorial

February 14, 2025

Formula

$$J_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- y_i are the true target values
- $\hat{y}_i$ are the predicted values
- n is the number of observations
- J_{MSE} denotes the MSE-based loss function (here we use J to represent any loss/cost function)

Best for

- **Penalizing large errors heavily:** Squaring the differences means that larger errors have a disproportionately higher impact on the overall cost.

Characteristics

- **Differentiable and easy to compute:** The MSE-based cost function is smooth and lends itself well to gradient-based optimization.
- **Sensitive to outliers:** Squaring amplifies large errors, making the metric more sensitive to outliers.
- **Encourages predictions close to the mean:** Especially when the target distribution is unimodal, the MSE criterion drives predictions toward the mean of the targets.

Gaussian Distribution

PDF of a Gaussian (Normal) distribution with mean μ and variance σ^2 for a variable x:

$$\mathcal{L}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where we use $\mathcal{L}$ to indicate the likelihood function.

Probabilistic Perspective (Derivation)

Assuming the prediction errors follow a Gaussian distribution with mean 0 and variance σ^2, the likelihood $\mathcal{L}$ for a single observation y_i given the predicted value $\hat{y}_i$ is:

$$\mathcal{L}(y_i \mid \hat{y}_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \hat{y}_i)^2}{2\sigma^2}\right).$$

Steps:

1. $\log \mathcal{L}(y_i \mid \hat{y}_i) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \log\left(\exp\left(-\frac{(y_i - \hat{y}_i)^2}{2\sigma^2}\right)\right)$
2. $\log \mathcal{L}(y_i \mid \hat{y}_i) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{(y_i - \hat{y}_i)^2}{2\sigma^2}$
3. $J_i = -\log \mathcal{L}(y_i \mid \hat{y}_i) = \frac{(y_i - \hat{y}_i)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$

Single-Point vs. Full Likelihood

- Usually, **likelihood** refers to the joint function of *all* data points.
- However, each individual term in the product (or sum in log space) is often called the likelihood contribution of a single observation.
- We can thus show the negative log-likelihood derivation *per data point*, then combine them for the full dataset.

Negative Log-Likelihood for the Entire Dataset

For n independent observations, the joint likelihood is the product of individual likelihoods:

$$J_{\text{NLL}} = -\log \mathcal{L}\big(\{y_i\} \mid \{\hat{y}_i\}\big) = \sum_{i=1}^{n} \Big[\frac{(y_i - \hat{y}_i)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)\Big].$$

Ignoring the constant term $\frac{1}{2} n \log(2\pi\sigma^2)$ gives:

$$J_{\text{NLL}} = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad + (\text{const.}).$$

Minimizing this w.r.t. $\hat{y}_i \implies$ **Minimizing MSE.**

Geometric Perspective (Derivation)

$$\|y - \hat{y}\|_2^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

- The MSE corresponds to squared Euclidean (L2) distance.
- Minimizing $\|y - \hat{y}\|_2^2$ means finding the point $\hat{y}$ closest to the actual target y.

Connection to Linear Regression

- MSE is the standard cost function in **Ordinary Least Squares (OLS)** regression.
- Linear model: $y = X\beta + \varepsilon$ where X is the design matrix, β is the parameter vector, and ε is noise.
- Minimizing $J_{\text{MSE}}(\beta) = \sum_{i=1}^{n}(y_i - X_i\beta)^2$ leads to the OLS estimate.
- Closed-form solution (assuming invertibility): $\hat{\beta} = (X^T X)^{-1} X^T y$.

Detailed Derivation of the OLS Solution

1. Cost function:

$$J_{\text{MSE}}(\beta) = \sum_{i=1}^{n} (y_i - X_i\beta)^2,$$

in matrix form:

$$J_{\text{MSE}}(\beta) = (y - X\beta)^T (y - X\beta).$$

2. Take the gradient w.r.t. β:

$$(y - X\beta)^T (y - X\beta) = y^T y - 2\, y^T X\beta + \beta^T X^T X\beta.$$

$$\nabla_\beta J_{\text{MSE}}(\beta) = -2X^T y + 2X^T X\beta.$$

3. Set the gradient to zero (Normal Equations):

$$X^T X\hat{\beta} = X^T y.$$

4. Solve for $\hat{\beta}$:

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

Summation-Based Derivation (Without Matrix Algebra)

Model:

$$\hat{y}_i = \beta_0 + \sum_{j=1}^{d} \beta_j \, x_{ij}.$$

MSE cost:

$$J_{\text{MSE}}(\beta_0, \beta_1, \ldots, \beta_d) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \Big(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\Big)^2.$$

Partial derivatives:

$$\frac{\partial J_{\text{MSE}}}{\partial \beta_0} = \sum_{i=1}^{n} -2\Big(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\Big) = 0,$$

$$\frac{\partial J_{\text{MSE}}}{\partial \beta_k} = \sum_{i=1}^{n} -2\Big(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\Big) x_{ik} = 0 \quad (k \geq 1).$$

Solve the resulting $(d+1)$**-equation system:** equivalent to the Normal Equation solution.

Gradient-Based Optimization

$$\frac{\partial J_{\text{MSE}}}{\partial \hat{y}_i} = \frac{2}{n}(\hat{y}_i - y_i).$$

- Because J_{MSE} is smooth and differentiable, it's well-suited for gradient-based methods.
- In parametric models (e.g., neural networks), apply the chain rule to backprop through parameters.

Relationship to $\mathbf{R^2}$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}.$$

- Reducing $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ increases R^2.
- A higher R^2 indicates a better fit of the model to the data.