Mean Squared Error (MSE)

Formula

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

where:

- 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.

Interpretations

Gaussian Distribution

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

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

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:

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

1. *Write out the PDF*:

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

2. *Take the logarithm*:

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

3. *Use log rules*:

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

4. *Multiply by -1 to define the negative log-likelihood* (our cost J):

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

Single-Point vs. Full Likelihood It s common to discuss a single data point s probability density as a likelihood in a derivation. Formally, the word likelihood typically refers to the joint function of all data points, but each individual term in that product (or each individual summand in the log-likelihood) can also be referred to as the likelihood contribution of a single observation. Thus, we can show the negative log-likelihood derivation on a per-data-point basis, then multiply (or sum in log space) to account for the entire dataset.

Negative Log-Likelihood for the Entire Dataset

For n independent observations, the joint likelihood is the product of individual likelihoods. Consequently, the negative log-likelihood for all observations becomes the sum:

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

Note on Curly Braces ($\{\}$): When we write $\{y_i\}$ or $\{\hat{y}_i\}$, we're indicating the entire set (or collection) of y_i or $\hat{y}_i$ values for $i = 1, \ldots, n$. This is standard mathematical notation for describing a group of elements, rather than a single element.

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

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

A Note on the $\frac{1}{2\sigma^2}$ Factor
From a purely optimization perspective (where σ^2 is fixed and we only optimize $\hat{y}$ or model parameters), this constant factor does not affect the location of the minimum. So, many treatments omit it when they only care about the solution for $\hat{y}$. However, when the variance σ^2 is also being optimized (as in a full maximum-likelihood approach for both mean and variance), we need to keep that factor to reflect the exact Gaussian log-likelihood.

Minimizing this with respect to $\hat{y}_i$ is equivalent to minimizing the sum of squared errors. Hence, **minimizing the negative log-likelihood** under these Gaussian assumptions is equivalent to **minimizing MSE**.

Geometric Perspective (Derivation)

The MSE can also be interpreted as the squared Euclidean (L2) distance between the predicted and true values:

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

Minimizing this distance is equivalent to finding the point $\hat{y}$ closest (in an L2 sense) to the actual target y.

Connection to Linear Regression

MSE is the standard cost function in Ordinary Least Squares (OLS) regression. In linear regression, the model is often expressed as:

$$y = X\beta + \varepsilon,$$

where X is the design matrix of input features, β is the parameter vector, and ε represents normally distributed noise. Minimizing the MSE cost:

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

yields the OLS estimates. If the columns of X are linearly independent, the closed-form solution is:

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

Detailed Derivation of the OLS Solution

Here is a step-by-step outline of how we arrive at the formula $\hat{\beta} = (X^T X)^{-1} X^T y$ in linear algebra terms:

1. **Set up the cost function:**
$$J_{\text{MSE}}(\beta) = \sum_{i=1}^{n}\big(y_i - X_i\beta\big)^2.$$
In matrix form, if y is the $n \times 1$ vector of targets and X is the $n \times d$ matrix of features (with each row corresponding to one observation), then:
$$J_{\text{MSE}}(\beta) = (y - X\beta)^T (y - X\beta).$$

2. **Take the gradient w.r.t. β:** We can expand the above cost:
$$(y - X\beta)^T (y - X\beta) = y^T y - y^T X\beta - (X\beta)^T y + (X\beta)^T (X\beta).$$
Notice that $y^T X\beta$ and $(X\beta)^T y$ are scalars (single numbers), so they are equal. Thus:
$$(y - X\beta)^T (y - X\beta) = y^T y - 2\, y^T X\beta + \beta^T X^T X\beta.$$
Now take the derivative (gradient) w.r.t. β:
$$\nabla_\beta J_{\text{MSE}}(\beta) = -2X^T y + 2X^T X\beta.$$
(We use basic rules of matrix calculus here, noting that the derivative of $\beta^T A\beta$ w.r.t. β is $2A\beta$ if A is symmetric. Here, $X^T X$ is symmetric.)

3. **Set the gradient to zero (the Normal Equations):** To minimize $J_{\text{MSE}}(\beta)$, we set its gradient to 0:
$$-2X^T y + 2X^T X\hat{\beta} = 0 \quad \Rightarrow \quad X^T X\hat{\beta} = X^T y.$$
This equation is known as the *Normal Equation*.

4. **Solve for $\hat{\beta}$:** Provided $X^T X$ is invertible (which requires that X has full column rank), we can multiply both sides by $(X^T X)^{-1}$:
$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

5. **Interpretation:**
   - $X^T X$ captures the correlations among features.
   - $X^T y$ captures how each feature relates to the target vector.
   - $(X^T X)^{-1} X^T y$ thus gives the coefficients that minimize the overall sum of squared errors.

This derivation relies on understanding how to take matrix derivatives and the condition that $X^T X$ be invertible. When these assumptions hold, the formula neatly expresses the solution that best fits the data in the least squares sense.

Summation-Based Derivation (Without Matrix Algebra)

While matrix notation is concise, we can also derive the same result using summations explicitly. Let s assume our model is:

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

where x_{ij} is the value of the j-th feature for the i-th data point, β_0 is an intercept term, and β_j are the feature coefficients.

Our goal is to minimize the MSE cost function:

$$J_{\text{MSE}}(\beta_0, \beta_1, \ldots, \beta_d) = \sum_{i=1}^{n} \bigl(y_i - \hat{y}_i\bigr)^2.$$

Substituting $\hat{y}_i$:

$$J_{\text{MSE}} = \sum_{i=1}^{n} \Bigl(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j \, x_{ij}\Bigr)^2.$$

1. Take partial derivatives w.r.t. each β_k For β_0:

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

(when set to 0 for the optimum). Simplify to get:

$$\sum_{i=1}^{n} \Bigl(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\Bigr) = 0.$$

For each β_k with $k \geq 1$:

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

2. System of $d+1$ equations These partial derivatives give us $d+1$ simultaneous equations (one for β_0, and one for each β_k, $k = 1, \ldots, d$). In compact form:

$$\begin{cases} \sum_{i=1}^{n} \bigl(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\bigr) = 0, \\ \sum_{i=1}^{n} x_{i1} \bigl(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\bigr) = 0, \\ \vdots \\ \sum_{i=1}^{n} x_{id} \bigl(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij}\bigr) = 0. \end{cases}$$

Solving this system yields the same result as applying the matrix-based Normal Equation: each equation here corresponds to a row in $X^T X \hat{\beta} = X^T y$.

Hence, whether we choose the summation approach (manually solving these $d+1$ equations) or the matrix approach (using linear algebra), we arrive at the same solution. The matrix form is simply a more compact and elegant representation of these summations.

Gradient-Based Optimization

Because J_{MSE} is smooth and differentiable, it fits perfectly in gradient-based methods. The partial derivative of J_{MSE} with respect to the prediction $\hat{y}_i$ is:

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

When using a parametric model, this gradient is propagated back through the model parameters (e.g., weights in neural networks) by the chain rule.

Relationship to R^2

Minimizing J_{MSE} is closely related to maximizing the coefficient of determination R^2, which measures how well the model explains the variance in the observed data. The R^2 value is defined (in one of its forms) as:

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

where $\bar{y}$ is the mean of the true targets y_i. Reducing $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ (i.e., the SSE) will increase R^2. A higher R^2 indicates a tighter fit of the model to the data.