# Loss Function

CT-466 | Week 4 - Lecture 8

Instructor: Mehar Fatima Shaikh

# Loss Function

- A loss function in a neural network evaluates how different the predicted output is from the actual target value.
- It measures the error and guides the model to improve its predictions using backpropagation and gradient descent.

# Types of Loss Functions

- Regression Loss Functions (for continuous outputs):
  - Mean Squared Error (MSE)
  - Mean Absolute Error (MAE)
  - Huber Loss (less sensitive to outliers)
- Classification Loss Functions (for discrete outputs):
  - Cross-Entropy Loss (Log Loss)
  - Hinge Loss (commonly used in SVMs)

# Mean Squared Error (MSE)

The formula for MSE is:

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

Where:

- n = number of data points
- $y_i$ = actual (true) value
- $\hat{y}_i$ = predicted value

# Mean Absolute Error (MAE)

The formula for MAE is:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

Where:

- n = number of data points
- $y_i$ = actual (true) value
- $\hat{y}_i$ = predicted value

# Why Squaring the Error Matters

- Small differences stay small after squaring (e.g., $2^2 = 4$)
- Large differences grow much bigger (e.g., $10^2 = 100$)
- So, MSE penalizes larger errors more heavily.

# Derivative of MSE

- For a single sample: $E = (y - \hat{y})^2$
- Taking derivative w.r.t $\hat{y}$: $\partial E / \partial \hat{y} = -2(y - \hat{y})$
- The derivative is a linear function of the error $(y - \hat{y})$.
- The gradient changes smoothly with error no jumps.
- Optimizers like gradient descent work efficiently with it.

# Why This Helps

- The derivative is a **linear function** of the error (y−y^).
- This means the gradient changes smoothly as the error changes.
- No jumps or discontinuities → the optimizer (like gradient descent) can take small, consistent steps toward minimizing error.

# Comparison with MAE

- MAE: $E = |y - \hat{y}|$
- Derivative:

- Not smooth — jumps from -1 to 1 — harder to optimize.

# Problem with Simple Average Error

- Example:

- (misleading!)
- $MAE = (|-5| + |+5|) / 2 = 5$
- Correctly shows average error magnitude.

# Comparison

| Mean Squared Error (MSE) | Mean Absolute Error (MAE) |
| --- | --- |
| Squares the errors before taking the average. | Calculates the absolute difference between actual and predicted values. |
| Gives more weight to larger errors compared to smaller ones. | Treats all errors uniformly, without emphasizing larger ones. |
| Provides smoother gradients, which helps in optimization. | Less affected by outliers compared to MSE. |

# Huber Loss

- Behaves like MSE for small errors, MAE for large errors.
- Formula:
- $$L(y, \hat{y}) = \begin{cases} \frac{(y - \hat{y})^2}{2} & \text{if } |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta) & \text{if } |y - \hat{y}| > \delta \end{cases}$$
- Where y = actual value, $\hat{y}$ = predicted value, $\delta$ = threshold

# Huber Loss Example

- Given:
- $N = 3$
- $Y = [5, 8, 3]$ (actual values)
- $\hat{Y} = [5.7, 7.8, 3.4]$ (predicted values)
- $\delta = 0.5$
- **Solution :**
- Errors = [0.7, 0.2, 0.4]
- Error $0.7 > \delta \rightarrow L = 0.5 \times (0.7 - 0.25) = 0.225$
- Error $0.2 \leq \delta \rightarrow L = 0.5 \times 0.2^2 = 0.020$
- Error $0.4 \leq \delta \rightarrow L = 0.5 \times 0.4^2 = 0.080$
- Average Huber Loss = $(0.225 + 0.020 + 0.080) / 3 \approx 0.108$
- Final Answer: Huber Loss $\approx 0.108$

# Hinge Loss

- Hinge Loss is used for training Support Vector Machines (SVMs) in classification tasks. The formula for Hinge Loss is:

$$L = \sum max(0, 1 - y_i * f(x_i))$$

# Hinge Loss Example

- Given:
- Email A: y = -1, f(x) = -2
- Email B: y = +1, f(x) = 0.6
- Email C: y = -1, f(x) = 0.3
- Calculations:
- Loss_A = max(0, 1 - (-1 * -2)) = max(0, 1 - 2) = 0
- Loss_B = max(0, 1 - (1 * 0.6)) = max(0, 0.4) = 0.4
- Loss_C = max(0, 1 - (-1 * 0.3)) = max(0, 1.3) = 1.3
- Total Hinge Loss = 0 + 0.4 + 1.3 = 1.7
- ***Final Hinge Loss: 1.7***

# Binary Cross-Entropy (Log Loss)

- Binary Cross-Entropy is used for binary classification. The formula is:

$$BCE = -(1/n) * \sum [y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)]$$

# Binary Cross-Entropy (Log Loss)

- Given:
- Email A: $y = 1$, $\hat{y} = 0.9$
- Email B: $y = 0$, $\hat{y} = 0.7$
- Email C: $y = 1$, $\hat{y} = 0.4$
- Calculations:
- $\text{Loss\_A} = -\log(0.9) \approx 0.1054$
- $\text{Loss\_B} = -\log(0.3) \approx 1.2040$
- $\text{Loss\_C} = -\log(0.4) \approx 0.9163$
- $\text{Total} = 0.1054 + 1.2040 + 0.9163 = 2.2257$
- $\text{BCE} = 2.2257 / 3 \approx 0.7419$
- ***Final Binary Cross-Entropy: 0.7419***

# Categorical Cross-Entropy

- Categorical Cross-Entropy is used for multi-class classification problems. The formula is:
- $CCE = -\sum y_i * log(\hat{y}_i)$

# Categorical Cross-Entropy

- Given:
- Data Point 1: Actual = X, $\hat{y} = 0.7$
- Data Point 2: Actual = Y, $\hat{y} = 0.8$
- Data Point 3: Actual = Z, $\hat{y} = 0.9$
- Calculations:
- Loss_1 = -log(0.7) ≈ 0.3567
- Loss_2 = -log(0.8) ≈ 0.2231
- Loss_3 = -log(0.9) ≈ 0.1054
- Total = 0.3567 + 0.2231 + 0.1054 = 0.6852
- CCE = 0.6852 / 3 ≈ 0.2284
- ***Final Categorical Cross-Entropy: 0.2284***