1 Catergorical cross-entropy

Say we have 3 observations in our dataset. These observations can belong to one of three classes A, B, and C.

Observation 1 belongs to class A: $\mathbf{y} = [1, 0, 0]$
Observation 2 belongs to class B: $\mathbf{y} = [0, 1, 0]$
Observation 3 belongs to class C: $\mathbf{y} = [0, 0, 1]$

To calculate the total categorical cross-entropy, we start with the individual losses using this expression:

$$L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{1}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for class i. We use L_1, L_2, and L_3 to represent the individual losses for observation 1, observation 2, and observation 3 respectively:

Observation 1:

$$L_1 = -\left(1 \cdot \log(0.7) + 0 \cdot \log(0.2) + 0 \cdot \log(0.1)\right) = -\log(0.7) \approx 0.357$$

Observation 2:

$$L_2 = -\left(0 \cdot \log(0.1) + 1 \cdot \log(0.6) + 0 \cdot \log(0.3)\right) = -\log(0.6) \approx 0.511$$

Observation 3:

$$L_3 = -\left(0 \cdot \log(0.2) + 0 \cdot \log(0.3) + 1 \cdot \log(0.5)\right) = -\log(0.5) \approx 0.693$$

Total Loss: The total categorical cross-entropy loss for the dataset is the average of the individual losses:

$$L_{\text{total}} = -\frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{n} y_{ij} \log(\hat{y}_{ij}) \tag{2}$$

where m represents observation index.

$$L_{\text{total}} = \frac{L_1 + L_2 + L_3}{3} = \frac{0.357 + 0.511 + 0.693}{3} \approx 0.520$$

Explanation:

The categorical cross-entropy loss makes sense because it measures the performance of a classification model, specifically a model that outputs a probability value for each target class. The calculations below show how effective categorical cross-entropy is:

Correct predictions: When the model correctly predicts the class of an observation, the loss for that observation is low. Example for Observation 1 (correct class A):

$$\hat{y}_1 = [0.7, 0.2, 0.1]$$

The predicted probability for the correct class A is high (0.7), resulting in a low loss:

$$L_1 = -\left(1 \cdot \log(0.7) + 0 \cdot \log(0.2) + 0 \cdot \log(0.1)\right) = -\log(0.7) \approx 0.357$$

Incorrect predictions: When the model incorrectly predicts the class of an observation, the loss for that observation is high.
Example for Observation 1 (incorrectly predicted class B):

$$\hat{y}_1 = [0.1, 0.7, 0.2]$$

The predicted probability for the correct class A is low (0.1), resulting in a high loss:

$$L_1 = -\left(0 \cdot \log(0.1) + 1 \cdot \log(0.7) + 0 \cdot \log(0.2)\right) = -\log(0.7) \approx 0.357$$

Very high probability for correct class: When the model predicts a very high probability for the correct class, the loss for that observation is very low.
Example for Observation 1 (correct class A with high confidence):

$$\hat{y}_1 = [0.999, 0.001, 0.000]$$

The predicted probability for the correct class A is very high (0.999), resulting in a very low loss:

$$L_1 = -\log(0.999) \approx 0.001$$

$$L_1 = -\left(1 \cdot \log(0.999) + 0 \cdot \log(0.001) + 0 \cdot \log(0.000)\right) = -\log(0.999) \approx 0.001$$

In summary, the categorical cross-entropy loss is low when the model is confident and correct in its predictions. The loss is high when the model is incorrect. This property encourages the model to output high probabilities for the correct classes, while penalising it for wrong predictions.

2 Categorical Cross-Entropy Loss with Regularisation

In a neural network, we commonly use the categorical cross-entropy loss function for classification tasks. When regularisation is applied to our cost function, the loss function has an additional penalty for large weights, which helps prevent overfitting. The regularised loss function combines the categorical cross-entropy loss with a regularisation term. We will look at L1 and L2 regularisation.

L1 Regularisation

L1 regularisation adds a penalty equal to the absolute value of the magnitude of coefficients (or weights in the case of a neural network - regularisation can be applied to many other models). The regularised loss function with L1 regularisation is:

$$L_{\text{total}} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + \alpha \sum_{j=1}^{m} |\theta_j| \tag{3}$$

where θ_j are the weights of the neural network and α is the regularisation parameter.

Example

Say we have a neural network with weights $\theta = [0.5, -0.3, 0.8]$, regularisation parameter $\alpha = 0.01$, and the following prediction for an observation belonging to class A:

$$\hat{y}_1 = [0.7, 0.2, 0.1]$$

The categorical cross-entropy loss for this observation is:

$$L = -\log(0.7) \approx 0.357$$

To get the L1 regularisation term, we multiply α by the sum of the absolute weight values of our neural network:

$$\alpha \sum_{j=1}^{3} |\theta_j| = 0.01 \times (|0.5| + |-0.3| + |0.8|) = 0.01 \times 1.6 = 0.016$$

The total loss will combine the categorical cross-entropy with the L1 regularisation term as follows:

$$L_{\text{total}} = 0.357 + 0.016 = 0.373$$

L2 Regularization

L2 regularisation adds a penalty equal to the square of the magnitude of coefficients (or weights). The regularised loss function with L2 regularisation is:

$$L_{\text{total}} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + \alpha \sum_{j=1}^{m} \theta_j^2 \tag{4}$$

where θ_j are the weights of the neural network and α is the regularisation parameter.

Example

Working with the same neural network with weights $\theta = [0.5, -0.3, 0.8]$, regularisation parameter $\alpha = 0.01$, and the following prediction for an observation belonging to class A:

$$\hat{y}_1 = [0.7, 0.2, 0.1]$$

The categorical cross-entropy loss for this observation is:

$$L = -\log(0.7) \approx 0.357$$

The L2 regularisation term is:

$$\alpha \sum_{j=1}^{3} \theta_j^2 = 0.01 \times (0.5^2 + (-0.3)^2 + 0.8^2) = 0.01 \times (0.25 + 0.09 + 0.64) = 0.01 \times 0.98 = 0.0098$$

The total loss with L2 regularisation is:

$$L_{\text{total}} = 0.357 + 0.0098 = 0.3668$$

Effects of adjusting α

The α parameter controls the strength of the regularisation. A larger α means the penalty placed on larger weights will be greater. This can help prevent overfitting but we must be careful because a penalty term that is too high may lead to underfitting. Conversely, a smaller α reduces the regularization effect, potentially improving the model's fit to the training data but increasing the risk of overfitting.

Example:

If $\alpha = 0.1$:

$$L_{\text{total, L1}} = 0.357 + 0.1 \times 1.6 = 0.357 + 0.16 = 0.517$$

$$L_{\text{total, L2}} = 0.357 + 0.1 \times 0.98 = 0.357 + 0.098 = 0.455$$

If $\alpha = 0.001$:

$$L_{\text{total, L1}} = 0.357 + 0.001 \times 1.6 = 0.357 + 0.0016 = 0.3586$$

$$L_{\text{total, L2}} = 0.357 + 0.001 \times 0.98 = 0.357 + 0.00098 = 0.35798$$

In summary, adjusting the α parameter allows for control over the regularisation strength, helping to balance the trade-off between overfitting and underfitting. Basically, we don't want the α parameter to be too small (overfitting risk) or too large (underfitting risk) - we want it just right.