Entropy Calculation in Classification Tasks

1. General Information

In machine learning classification tasks, a confusion matrix is used to evaluate the performance of a model. It shows the actual class labels vs. the predicted labels, helping to determine how well a model can distinguish between different classes.

Entropy, a concept from information theory, measures uncertainty or disorder. In the context of a confusion matrix, entropy quantifies how balanced the misclassifications are between pairs of classes. Higher entropy indicates more balanced misclassification between classes, which often suggests that the model has sufficient training data for both classes.

2. Mathematical Formula and Explanation

The entropy between two classes i and j based on their misclassifications in a confusion matrix is calculated using the following formula:

$$e_{ij} = -\left[\frac{C_{ij}}{C_{ij}+C_{ji}}\ln\left(\frac{C_{ij}}{C_{ij}+C_{ji}}\right)+\frac{C_{ji}}{C_{ij}+C_{ji}}\ln\left(\frac{C_{ji}}{C_{ij}+C_{ji}}\right)\right]$$

Where:

- C_{ij} is the number of times class i was predicted as class j (misclassification).
- C_{ji} is the number of times class j was predicted as class i.
- $\ln$ is the natural logarithm.

The entropy formula calculates the degree of uncertainty in the model's confusion between two classes. Higher entropy implies that the model is equally likely to confuse the two classes, indicating a balanced dataset for those classes.

3. Result Interpretation

Higher Entropy: Suggests that the misclassifications between two classes are evenly distributed. This often indicates that the model has sufficient training data for both classes.

Lower Entropy: Implies that the model is more likely to misclassify one class as the other, indicating possible class imbalance or insufficient training data.

In this project, you'll calculate the entropy for pairs of classes in a confusion matrix, compare the values, and determine which class pair has the most balanced confusion, implying sufficient training data.

4. Problem and Solution

Given the following confusion matrix for a 3-class classification problem (with classes A, B, and C): which pair of classes are we more likely to have sufficient training data?

$$CM = \begin{bmatrix} 0.65 & 0.15 & 0.20 \\ 0.10 & 0.70 & 0.20 \\ 0.15 & 0.10 & 0.75 \end{bmatrix}$$

The entropy values for each pair of classes, based on the given confusion matrix, are as follows (Please see the Python Code):

Entropy for pair (A, B): 0.6730

Entropy for pair (A, C): 0.6829

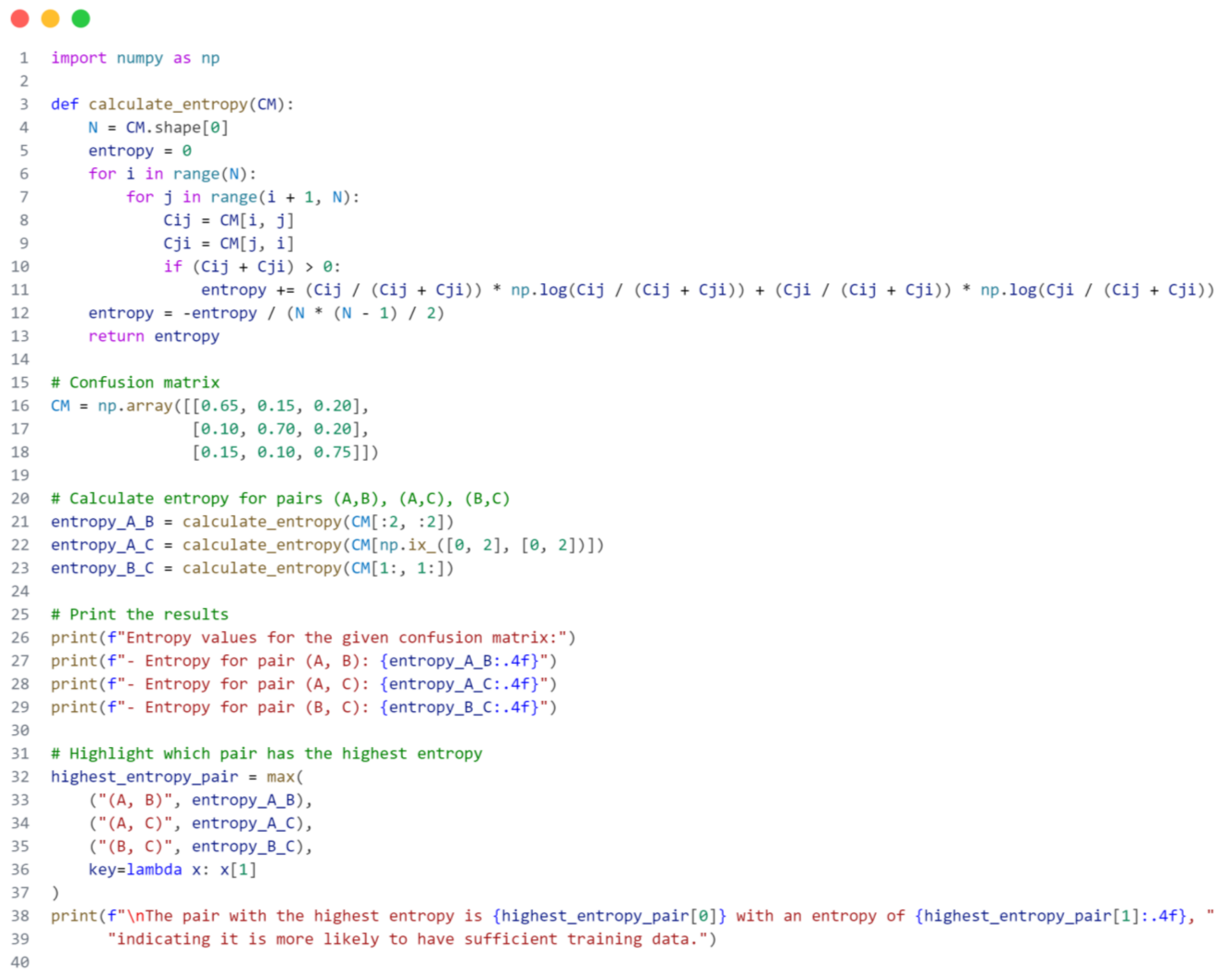
Entropy for pair (B, C): 0.6365

Class Pair (A, C) has the highest entropy at 0.6829. This indicates that the model is equally confused between class A and class C, suggesting that the training data for these classes is balanced and likely sufficient. The model struggles equally with distinguishing between these two classes, which often implies the model has been trained with enough representative data for both classes.

Class Pair (A, B) has a slightly lower entropy at 0.6730, meaning that the model is also reasonably balanced in confusing class A and class B, but there may be a slight imbalance in the data compared to class pair (A, C).

Class Pair (B, C) has the lowest entropy at 0.6365, suggesting that there is more imbalance in the misclassifications between these two classes. The model is more likely to confuse one class with the other, possibly due to insufficient or imbalanced training data for either class B or class C.

5. Python Code



```python
import numpy as np

def calculate_entropy(CM):
    N = CM.shape[0]
    entropy = 0
    for i in range(N):
        for j in range(i + 1, N):
            Cij = CM[i, j]
            Cji = CM[j, i]
            if (Cij + Cji) > 0:
                entropy += (Cij / (Cij + Cji)) * np.log(Cij / (Cij + Cji)) + (Cji / (Cij + Cji)) * np.log(Cji / (Cij + Cji))
    entropy = -entropy / (N * (N - 1) / 2)
    return entropy

# Confusion matrix
CM = np.array([[0.65, 0.15, 0.20],
               [0.10, 0.70, 0.20],
               [0.15, 0.10, 0.75]])

# Calculate entropy for pairs (A,B), (A,C), (B,C)
entropy_A_B = calculate_entropy(CM[:2, :2])
entropy_A_C = calculate_entropy(CM[np.ix_([0, 2], [0, 2])])
entropy_B_C = calculate_entropy(CM[1:, 1:])

# Print the results
print(f"Entropy values for the given confusion matrix:")
print(f"- Entropy for pair (A, B): {entropy_A_B:.4f}")
print(f"- Entropy for pair (A, C): {entropy_A_C:.4f}")
print(f"- Entropy for pair (B, C): {entropy_B_C:.4f}")

# Highlight which pair has the highest entropy
highest_entropy_pair = max(
    ("(A, B)", entropy_A_B),
    ("(A, C)", entropy_A_C),
    ("(B, C)", entropy_B_C),
    key=lambda x: x[1]
)
print(f"\nThe pair with the highest entropy is {highest_entropy_pair[0]} with an entropy of {highest_entropy_pair[1]:.4f}, "
      "indicating it is more likely to have sufficient training data.")
```

Gulumjanly, Z. (2024). *Confusion Matrix Entropy* . GitHub.
https://github.com/ziraddingulumjanly/ConfusionMatrixEntropy