Origins of The Cross-Entropy Loss Function

Gaining Intuition on the Popular Loss Function

Ibrahim Habib

2025-04-16

Table of contents

The cross-entropy loss function is one of the most widely used loss functions in deep learning, particularly for classification tasks. I've used it before and understood the math behind it. However, when I studied it I didn't understand how this function came or why was it in this particular form. It is clear that using it will penalize incorrect predictions, but still, it wasn't clear why this exact form.

In this post, I will attempt to explain the meaning behind cross-entropy loss and show its origins. Hopefully, by the end of reading this, you will have a better understanding of cross-entropy. So grab your favorite cup of coffee and let's get started!

1 Surprise and Information Theory

To understand cross-entropy we need to visit the field of science from which it originates: Information Theory. Information theory is the field of science concerned with quantifying and communicating information. It was founded by Claude Shannon in the 1940s and today it is used in many fields including deep learning.

We'll begin our discussion by introducing an information theory concept called **surprise**.

1.1 Explanation and Intuition

To understand surprise we will visit the problem of compressing data. Let's say that we have a large stream of data and we want to use a smaller version of it to save storage and transmission costs. How hard is a stream of data to compress?

Imagine a stream of data where each token has the same exact value. Well, this is very easy to compress as all we need to do is store the value and the number of tokens. This is a very boring stream that doesn't contain any *surprises*.

Now imagine the opposite case: a stream where each token has a different value. This one is extremely difficult to compress as we need to store each token separately. Given the history of the tokens seen, the next token always *surprises* us.

Armed with these two examples, we can now revisit the question. A stream with a lot of unexpected tokens, or surprises, is hard to compress. Information theory quantifies the "unexpectedness" of a token in a term called **surprise**.

1.2 Mathematical Definition

Let's now see how Claude Shannon defined surprise mathematically. Let's say you have an event x with a probability $P(x)$. Shannon defined the surprise of this event as:

$$S(x) = \log_2\left(\frac{1}{P(x)}\right) = -\log_2(P(x))$$

Note that the more likely an event x is, the larger is $P(x)$, and hence, by definition, the smaller is $S(x)$. In other words, the more likely an event is, the less surprising it is, and vice versa.

This indeed follows our intuition on the meaning of an event's surprise.

```python
import numpy as np
import pandas as pd
from tabulate import tabulate
from IPython.display import display, Markdown

dice_sum_probabilities = np.linspace(0.01, 1, 10) # Don't start at 0 because its surprise gro

def calc_surprise(x):
  if x == 0 or x == 1:
    return 0
  else:
    return -np.log2(x)

surprises = [calc_surprise(p) for p in dice_sum_probabilities]

table = pd.DataFrame({
    'Probability': dice_sum_probabilities,
    'Surprise': surprises
})

display(Markdown(table.to_markdown(index=False, floatfmt=".2f")))
```

Probability	Surprise
0.01	6.64
0.12	3.06
0.23	2.12
0.34	1.56
0.45	1.15
0.56	0.84
0.67	0.58
0.78	0.36
0.89	0.17
1.00	0.00

Caution

Note that the case where $P(x) = 0$ must be handled carefully because the surprise grows infinitely large. An event that never happens is extremely surprising.

1.3 Example

Consider the classic classroom example of throwing two dice and observing the sum of their values. The following table shows the possible sums, their probabilities, and their surprises:

```python
dice_1 = list(range(1, 7))
dice_2 = list(range(1, 7))

sample_space = [(x, y) for x in dice_1 for y in dice_2]
dice_sum = [x + y for x, y in sample_space]

outcomes = list(range(2, 13))
dice_sum_probabilities = [round(dice_sum.count(outcome) / len(sample_space), 2) for outcome 
surprises = [round(calc_surprise(p), 2) for p in dice_sum_probabilities]

table = pd.DataFrame({
    'Sum': outcomes,
    'Probability': dice_sum_probabilities,
    'Surprise': surprises
})

display(Markdown(table.to_markdown(index=False)))
```

Sum	Probability	Surprise
2	0.03	5.06
3	0.06	4.06
4	0.08	3.64
5	0.11	3.18
6	0.14	2.84
7	0.17	2.56
8	0.14	2.84
9	0.11	3.18
10	0.08	3.64
11	0.06	4.06
12	0.03	5.06

As we would expect, sums closer to the center (7), which are more expected, have lower surprises compared to those on the edges (2 and 12).

2 Entropy

We're now ready to introduce the second information theory concept discussed here: **Entropy**. While surprise measures the difficulty of compressing a single event, entropy measures the difficulty of compressing a stream of events. It is closely related to surprise. Actually, it is defined as the expected value of surprise.

2.1 Mathematical Definition Derivation

Let's now derive its definition mathematically. From your STAT101 class, you probably remember the definition of the expected value of a discrete random variable X:

$$E[f] = \sum_{x \in X} P(x) \cdot f(x)$$

As said before, entropy, denoted by $H(X)$, is the expected value of surprise. So by definition, we have:

$$H(P) = E[S] = \sum_{x \in X} P(x) \cdot S(x)$$

By substituting the definition of surprise we get the common definition of entropy:

$$H(P) = -\sum_{x \in X} P(x) \cdot \log_2 P(x)$$

You can easily get the definition of entropy for a continuous random variable by replacing the summation with an integral.

2.2 Example

To get a feel for entropy we'll consider the following examples:

1. A fair coin. (Bernoulli distribution)
2. A biased coin with a probability of heads of 0.9.
3. A biased coin with a probability of heads of 1.0 (deterministic).
4. The sum of two dice. (Our previous example)

Between the three coins, we should expect the fair coin to have the highest entropy because it is the most uncertain. The biased coin with a probability of heads of 0.9 should have a lower entropy, and the deterministic coin should have the lowest entropy of all three.

The following table shows the entropies of these distributions and confirms our understanding:

```
def calc_entropy(probabilities):
  return -sum(p * np.log2(p) for p in probabilities if p > 0) # Once again, 0s are handled di


fair_coin_entropy = calc_entropy([0.5, 0.5])
biased_coin_entropy = calc_entropy([0.9, 0.1])
deterministic_coin_entropy = abs(calc_entropy([1, 0])) # Abs to avoid -0.0
dice_sum_entropy = calc_entropy(dice_sum_probabilities)

table = pd.DataFrame({
    'Coin': ['Fair', 'Biased (p=0.9)', 'Deterministic', 'Dice Sum'],
    'Entropy': [fair_coin_entropy, biased_coin_entropy, deterministic_coin_entropy, dice_sum_
})
display(Markdown(table.to_markdown(index=False, floatfmt=".2f")))
```

Coin	Entropy
Fair	1.00
Biased (p=0.9)	0.47
Deterministic	0.00
Dice Sum	3.30

3 Cross-Entropy

We're now ready to face the beast itself: **Cross-Entropy**.

Entropy measured the average surprise of someone who knows the true distribution of the data and shaped his surprise (or compression technique) upon this knowledge. Earlier when we took the expected value of surprise, we used the true probability of the data to calculate the surprise.

But what if we're wrong? The surprise is based on our assumption of the distribution of the data which may be incorrect. This is where cross-entropy comes.

3.1 Explanation and Definition

Note that in the definition of entropy we used $P(x)$ twice: once to compute our surprise and once to weight it. The probability in the surprise comes from our knowledge and assumptions abot the data distrbution, the one upon which we build our expectations and, consequently, our surprise. The second probability, however, is the true distribution of the data. This is a very important distinction because our thoughts about the data may not be correct. The two probabilities may be different.

To simplify our discussion let's denote the true distribution of the data $P_{\text{true}}(x)$ and our assumptions about the data $P_{\text{assumed}}(x)$.

Let's now revisit our earlier definitions and use the new notation. The surprise or our difficulty in compressing a single event is given by and will be determined by our assumptions about the data. We don't have the true data distribution yet and are forced to shape our expectations based on the limited knowledge we have. So the surprise of an event x is given by:

$$S(x) = -\log_2 P_{\text{assumed}}(x)$$

Entropy, however, is the expected value of surprise. The number of times an event occurs is determined by reality and not our hopes. So entropy will be the average of the surprise weighted by the true distribution of the data:

$$H(P_{\text{true}}, P_{\text{assumed}}) = -\sum_{x \in X} P_{\text{true}}(x) \cdot \log_2(P_{\text{assumed}}(x))$$

That definition above is the definition of cross-entropy. It is the expected value of surprise calculated using the distribution P_{assumed} where each event actually occurs with a probability of P_{true}.

To write it in the common form, we call $P_{\text{assumed}}(x)$ as $Q(x)$ and $P_{\text{true}}(x)$ as $P(x)$. So the cross-entropy is given by:

$$H(P, Q) = -\sum_{x \in X} P(x) \cdot \log_2(Q(x))$$

Repeating again in different words, P is the distribution of the data generation process and Q is the distribution we are using to model the data and shape our expectations.

Note

Note that entropy is a special case of cross-entropy where the two distributions are the same and we know the data generation process.

3.2 Example

Let's get back to our coins. Let's say that you have a biased coin with a probability of heads of 0.7 You know that the coin is biased and you know the true probability of heads, but you are going to keep it a secret from your friends.

You gather 6 friends, give each of them the coin for a day, and ask them to guess the probability of heads. Each of them guesses a different value. Particularly, they guess the following values:

Friend	Probability of Heads
1	0
2	0.2
3	0.5
4	0.7
5	0.9
6	1

You are going to buy a gift for the friend whose expected distribution minimizes the cross-entropy So you calculate the cross-entropy for each of them.

```
def calc_cross_entropy(p, q):
  # Replace all 0s in q with 1e-10 to avoid log(0)
  q = np.clip(q, 1e-10, None)
  return -sum(p_i * np.log2(q_i) for p_i, q_i in zip(p, q))

true_distribution = [0.7, 0.3]

friends_expectations = [0, 0.2, 0.5, 0.7, 0.9, 1]
friends_cross_entropy = [calc_cross_entropy(true_distribution, [p, 1 - p]) for p in friends_

table = pd.DataFrame({
    'Friend': [f'Friend {i+1} (p={p})' for i, p in enumerate(friends_expectations)],
    'Cross Entropy': friends_cross_entropy
})

display(Markdown(table.to_markdown(index=False, floatfmt=".2f")))
```

Friend	Cross Entropy
Friend 1 (p=0)	23.25
Friend 2 (p=0.2)	1.72

Friend	Cross Entropy
Friend 3 (p=0.5)	1.00
Friend 4 (p=0.7)	0.88
Friend 5 (p=0.9)	1.10
Friend 6 (p=1)	9.97

We see that friend 4 who guessed the true distribution has the lowest cross-entropy. This example shows that the cross-entropy is a measure of how well our assumptions about the data match the true distribution of the data. The closer our assumptions are to reality, the lower the cross-entropy.

Now wait a minute! This is exactly what we want to do in deep learning. We create a model that makes assumptions about the data and generates predictions that we want to match the true distribution of the data. Cross-entropy tells us how close our predictions are to true distribution (training data) and we can use it as a loss function to train our model. The lower the cross-entropy, the better our model is at predicting the true distribution of the data.

4 Conclusion

In this post, we discussed the cross-entropy. We dived into information theory and the work of Claude Shannon to understand the origins of cross-entropy. We discussed the concepts of surprise and entropy and how they relate to cross-entropy. We also discussed how cross-entropy can be used as a loss function in deep learning. To summarize, we learned that cross-entropy is a measure of how well our assumptions about the data match the true distribution of the data. The closer our assumptions are to reality, the lower the cross-entropy.

I hope that you benefited from this post and that you now have a better understanding of cross-entropy. If you have any questions or comments, please feel free to reach out to me on X / Twitter, LinkedIn, or Email. Thanks for reading!