Entropy, Cross Entropy, KL Divergence: deep dive

The Cross Entropy cost function is omnipresent in Machine Learning as many problems involve matching probability distributions.

Although not strictly necessary, it may be interesting to explore this concept more deeply.

TL;DR

- Cross Entropy measures the "distance" between two distributions
- You will often come across Cross Entropy and KL Divergence in your future ML endeavours.

Entropy

In order to understand *Cross* Entropy, we first try to explain *Entropy*.

- A measure of the randomness of a distribution
- Interpreted as the number of bits of information obtained from a single sample
- an alternative to Gini for deciding which variable/what threshold to use in constructing a Decision Tree

What is a bit ?

- Amount of information need to reduce uncertainty by a factor of 2

Example: what is a bit

Here, a bit is the amount of information **not** a length of the message

Simple case: two equally probable outcomes

Two outcomes: Rain, Sun

Rain	$.5$
Sun	$.5$

Uncertainty about Rain is .5

If I tell you that it will Rain (so p_{Rain} goes from 0.5 to 1) the uncertainty is reduced by a factor of

$$\frac{1}{.5} = 2$$

$\log_2(2) = 1$

so this conveyed 1 bit of *information* regardless of how many *physical* bits were sent.

Definition of a bit: Uncertainty reduction of a factor of 2

One bit reduces uncertainty ($1/p$) by a factor of 2.

$\log_2(1/p) = -\log_2(p)$

Simple case: 4 equally probable outcomes

4 outcomes: Rain, Sun, Clouds, Drizzle

Rain	.25
Sun	.25
Clouds	.25
Drizzle	.25

Uncertainty about Rain is .25.

If I tell you that it will Rain (so p_{Rain} goes from 0.25 to 1) I've reduced the uncertainty by a factor of

$$\frac{1}{.25} = 4$$

This is $-\log_2(.25) = 2$ bits of information.

Outcomes with arbitrary probabilities

Two outcomes with unequal probability: Rain, Sun

$$\begin{array}{ll} \text{Rain} & .25 \\ \text{Sun} & .75 \end{array}$$

If I tell you that it will Rain (so $p_{\text{Rain}} = 1$) I've reduced the uncertainty by

$$\frac{1}{.25} = 4$$

so this is $\log_2(2) = 2$ bits of information.

If I tell you that there will be Sun (so $p_{\text{Sun}} = 1$) I've reduced the uncertainty by

$$\frac{1}{.75}$$

so this is $-\log_2(0.75) = 0.415$ bits of information.

In this case, the number of bits I convey to you is different depending on the message.

Average number of bits in a sample

The average number of bits (*information*) in a sample randomly drawn from the distribution is

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k})$$

For the 75%/25% case above

$$H_i = 0.75 * 0.415 + 0.25 * 2 = 0.81$$

25% of the time I give you 2 bits of information; 75% of the time only 0.415 bits

This is also called the **entropy**

It measures the unpredictability of the distribution.

A pure distribution has 0 entropy

Since $p_{i,k} = 1, log(p_{i,k}) = 0$

Cross entropy, KL Divergence

Whereas Entropy measured the randomness of a single distribution, Cross Entropy measures the "distance" between two probability distributions p and q.

We will call p the *true* distribution and q the estimated distribution.

Cross entropy between two discrete probability distributions p and q:

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

Similar to Entropy interpretation as weighted sum of bits but

- uses true probability p as weights
- uses estimated probability q to measure the bits

If distributions $p(x), q(x)$ are the same

- cross entropy equals entropy: $H(p, p) = H(p)$

The difference $(H(p, q) - H(p))$ is called the *Kullback-Leibler* or KL divergence.

Aside

There is another interpretation of Cross Entropy.

We've seen that the amount of information depends on number of bits of each outcome k: $\log_2(p_{i,k})$.

$H(p, q) = -\sum_x p(x) \log q(x)$ is the average number of bits if information is given by q

If the number of bits is different than $\log_2(p_{i,k})$

- then this message q is *optimal* in length
    - only if the distribution is some other $q_{i,k}$

The physical length will equal the information content only if $p(x) = q(x)$

In [3]:
```
print("Done")
```

```
Done
```