**Why do We use Cross-entropy in Deep Learning — Part 1**

**Explanation of one of the most widely used loss functions in Artificial Neural Networks**

If you’ve just started in the field of Deep Learning and have read some specialized articles, I am very sure that you have come across any of the following terms: *entropy*, *cross-entropy*, *binary cross-entropy,* or *categorical cross-entropy*.

All of them derive from the same concept: **Entropy**, which may be familiar to you from physics and chemistry. But, What does physics have to do with Artificial Intelligence? Where do the formulas come from? How do we interpret them? and, On which problems do we apply them?

Entropy, Cross-entropy and Categorical Cross-entropy are crucial concepts in the field of AI. However, not many courses or articles explain the terms in-depth, since it requires some time and mathematics to do it correctly. In this series of two posts, I’ll try to give a clear explanation using the resources and intuitions I gathered when I was looking for an article like this one. Furthermore, I’ll provide a formal mathematical description of the formulas and where they derive from.

**Note**. During my bibliographic research, I found [**this**](https://colah.github.io/posts/2015-09-Visual-Information/) incredible article by Christopher Olah, which inspired this article. If you have the time, I highly recommend reading it as well as checking Christopher’s website.

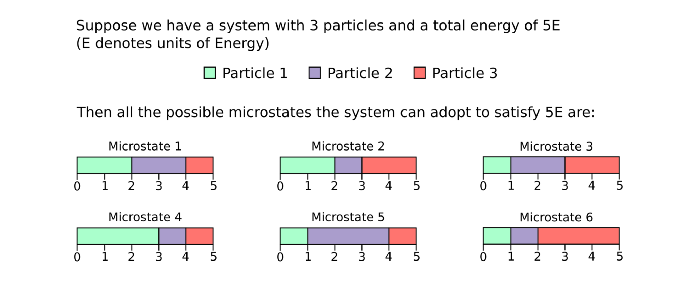
**Entropy definition**

The term **Entropy** might be familiar to you from your thermodynamic lessons in elementary physics and chemistry, but it has also two other conceptions from different branches of mathematics and computer science:

**Entropy in physics**

According to classical thermodynamics, *“****Entropy is a measure of the amount of disorder in a system****”*.

A physical system can be seen as a set of particles, each with an energy associated. Disorder refers to the number of possible configurations the system can adopt (also called *microstates*), the more configurations a system have, the greater the disorder and the greater the entropy.



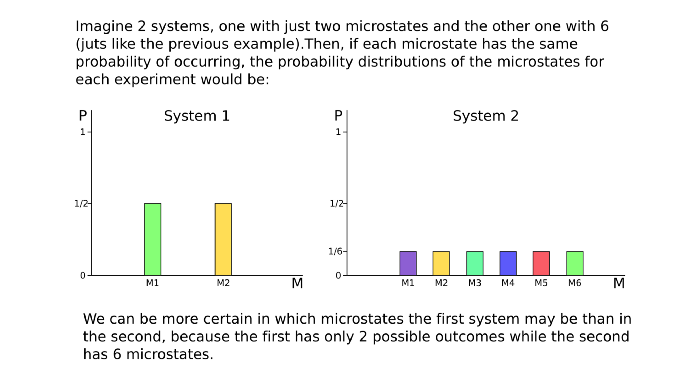
All possible microstates of a 3-particle system with overall energy 5E. The bars represent the energy associated with each particle, such that for Microstate 1: Particle 1 has 2E, Particle 2 2E and Particle 3 1E. Thus the total system energy is the sum 2E + 2E + 1E = 5E. Image by the author.

**Note1**. If you’d like to learn more about entropy in physics watch [this video](https://youtu.be/mg0hueOyoAw)

**Entropy in statistics**

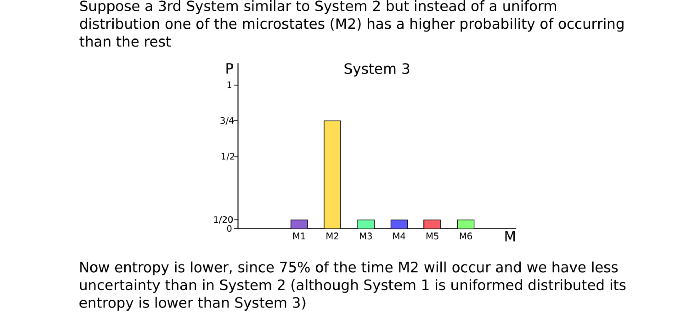
Deriving from the thermodynamics definition of entropy, we can also think of entropy as: ***“Entropy is a measure of the uncertainty of a random variable”***.

Let’s put it this way: our random variable is the microstates (possible energy configurations) of a system, where each configuration is a value our random variable can take. Then, assuming that each microstate has the same probability of happening, the more configurations our system can adopt the more uncertainty we have about what microstate the system is actually in.



Probability distributions for two systems: System 1 with two Microstates (M1, M2) and System 2 with six (M1, …, M6). Image by the author.

Another possible understanding of entropy in statistics could be: ***“Entropy is the amount of information of a random variable”***. Because the more uncertain we are about the outcome of a variable, the more information that variable is providing us. If we were sure or practically sure of the result of a variable, it would be worthless to measure it, as it wouldn’t be providing us with relevant information.



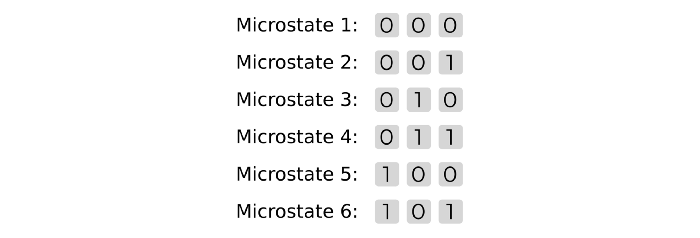
A 3rd System similar to System 2 but with lower entropy. Recall that all probabilities must ass up to 1. Image by the author.

Therefore, the higher the uncertainty, the higher the information, and the higher the entropy.

**Entropy in information theory**

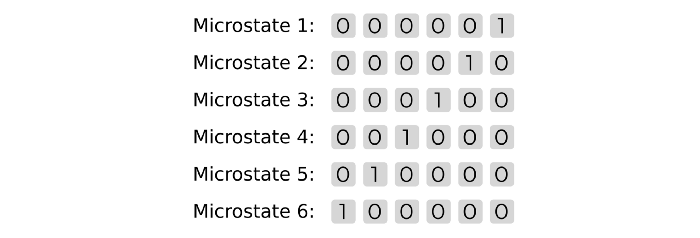
Information theory uses the concept of entropy to explain a very different concept than physics, ***“Entropy is the optimal average length for an encoded message”*.**

Continuing with the microstates example, suppose we would like to send the results of our experiment from our laboratory located in Madrid (Spain) across the world to the sunny beaches of Sydney (Australia).   
First, we have to decide which code system are we going to use for writing our message (binary, hexadecimal, ASCII, …) Suppose we choose binary code for the simplicity of the example.  
Next, we should define how we are going to represent each experiment result. Since we have 6 microstates, we could represent them as:



Binary representation of the 6 microstates. Image by the author

Another possible codification could be *one-hot encoding*:



Another possible encoding. Image by the author.

And there are many other possible solutions… **But which one is the optimal one?**   
Imagine now that each of the message’s bits has a price, let’s say 10€ per bit, now our first codification has an average price of 30€ per experiment, whereas one-hot encoding is 60€/experiment. Therefore, we naturally want to optimize our message so that the overall price is as minimum as possible.

**Note2**. Note that I am using the word “price” instead of “cost”. This is because I will later use the term cost to refer to another concept and in this way, I avoid generating ambiguity.

The optimal length to represent each microstate resulting from our experiments is what the entropy represents. **We cannot get a shorter codification for our message under the entropy barrier**.

**Entropy Formula**

If you look up the internet for the entropy formula, you might get a different result depending on the field you are searching for, and the reason is that many entropy formulas exist (watch this).

However, in Deep Learning we are interested in the **Shannon Entropy**, which comes from the Information Theory field (Shannon Entropy is also used in the statistical understanding of entropy)   
***Why do we use it?***The reason will come in the next post under the section [Cross-Entropy in Deep Learning](http://a), but first, let’s see where the formula derives from.

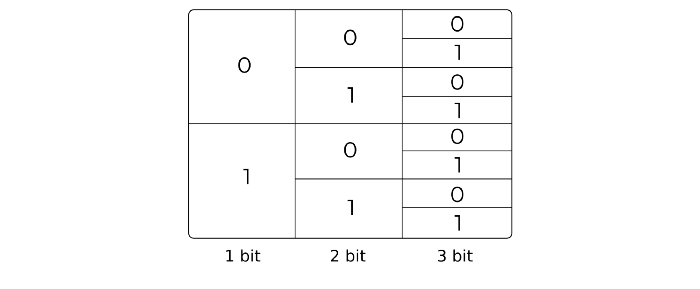
**Note3**. The reasoning to get to the entropy formula is based on Christopher Olah’s article I mentioned at the beginning of this post [[1]](https://colah.github.io/posts/2015-09-Visual-Information/). To get a deeper understanding I recommend you visit his website.

**The Space of Codewords**

The Space of Codewords refers to the set of all possible encoded messages you can create with *n* bits of information.

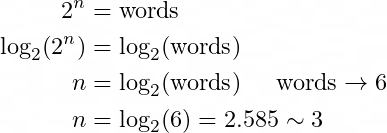
So for just 1 bit, you can only represent two different ***words*** (experiment results in our example): *0* and *1*. With 2 bits, you can express four words: *00*, *01*, *10* and *11*. And so on (see table).

Then, for the binary code, we can show that **with *n* representation bits we can express *2^n* words** (The demonstration is trivial, just think of the number of bits as gaps to fill, for the first gap you have 2 options (*0* or *1*), for the second two more options again (*0* or *1*), and so on for the rest of the bits. Now, to calculate all the options you have, you multiply *2 · 2 · 2 · … · 2 = 2^n* options (binary strings) to form with *n* bits)+



Space of Codewords for 3 bits. The last column shows the number of possible combinations (8), starting from the top: 000, 001, … Image by the author.

If we operate and clear n (number of bits), we can see that to represent 5 words in binary we need to use 3 bits.



Clearing n and computing the minimum number of bits needed to express the 6 experiments (we cannot have 2.585 bits so we must round up). Image by the author.

But, this only holds if we use the same number of bits to represent every word. What would happen then if we use a **variable length code**?

Recalling our encoded experiment results, we still have to figure out the cheapest way to send the messages. So instead of having: *000*, *001*, *010*, *011*, *100, 101;* we might try with the representation *00*, *01*, *100*, *101*, *110, 111*.

Now the average price of the new encoding system has decreased to ~26.67€/experiment instead of 30€.

And even more important, what if each of experiments 100, 101, 110 and 111 occurs only 5% of the time? Then the average message price would be lower since our messages will usually be formed by 2-bit words.

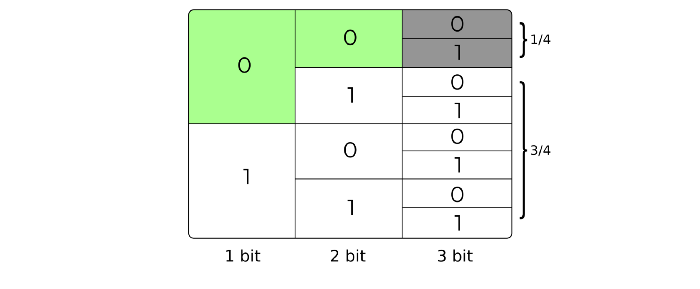
https://miro.medium.com/max/398/0*aO5qloEAscvBfIFT.png

Average Price of variable length encoding system. 80% of the time we’ll use 20€ words whereas only 20% the 30€ words. Image by the author.

**The Cost Function**

In order to achieve an optimal solution, **we need an objective function to minimize**.

(Remember when I said that I used the word price instead of cost because I wanted to introduce another different term and not create ambiguity? Now it’s the time) We define the ***cost*** as the portion of the Space of Codewords that we sacrifice when coding a word with n bits. When I state that the cost of using 2 bits to represent a word is 1/4, that means that I am losing 1/4 of my Space of Codewords.



Using 00 to represent a word prevents the use of all strings beginning with 00 (grey cells). This translates into a cost of 1/4. Image by the author

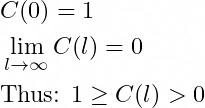
The objective function (from now on I will refer to it as the***Cost function***) must express the cost of using a code with a length of X bits to express a result (i.e. the function takes the length in bits of a code and outputs the cost of using that code in terms of the Space of Codewords)

Looking again at the table representing the Space of Codewords and the cost of using variable length codes, you might be able to get the following cost function:

https://miro.medium.com/max/100/0*aicBlKpEzd668B8o.png

Cost function for using x bits to encode a word. The result is expressed as a portion of the Space of Codewords. Image by the author

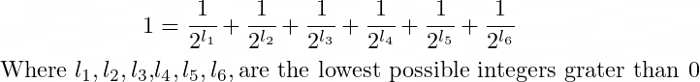
Analyzing the properties of the cost function *C(l):*



The Cost Function is bounded between 0 and 1. A code of length 0 sacrifices the whole Space of Codewords since we are sending an empty message which can just represent one result. Image by the author

**Achieving the minimum**

Finally, we must decide which is the optimal length to represent a word depending on the cost. We have the whole Space of Codewords to work for us, thus our budget is 1, and we have to find the shortest possible codes such that:



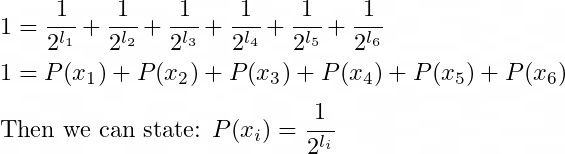
The function that defines our problem. Image by the author

Recall from previously that we are interested in using a **variable length code depending on the probabilities of each result to occur** (thus representing l*1, …, l6* as different variables and not just l)

Intuitively, we want to assign a **shorter** code (and therefore a higher cost in the Space of Codewords) to the most frequently occurring words, and a **longer** code (lower cost) to words that are less likely to appear. **But how do we quantify *shorter* and *longer*?** Is it 3 bits short or long?

The answer is Probability

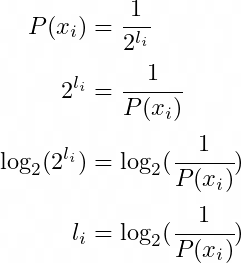
Think of it this way: We have a Cost function that must add up to 1 (otherwise we are wasting Space of Codewords) and we want to assign to each of the terms a cost proportional to their probability that, **add up to one!** Same as the cost function, the sum of the probabilities of the experiments is 1.



The lowest cost of a word of length L is the probability of occurrence of the word itself P(x) Image by the author

**Note4**. Although the statement I’ve just made needs mathematical proof, I don’t want to overload this post. If you are interested in reading it, [here](https://colah.github.io/posts/2015-09-Visual-Information/#optimal-encodingshttps://colah.github.io/posts/2015-09-Visual-Information/#optimal-encodings) is a link to a very intuitive and visual explanation.

Finally, we just have to operate the terms and write the length *l* as a function of the probability *P(x)*.

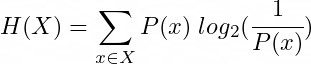


Length of a coded word as a function of its probability. Image by the author.

**Note5**. When you substitute the probability value (continuous in [0,1]) into the formula above, you will see that the result of length “l “ is a decimal value. But that can’t be right! How can we construct a code with 1.6145 bits? Well, I’ll leave that explanation to [Christopher](https://colah.github.io/posts/2015-09-Visual-Information/#fractional-bits) but the short answer is, we can’t, we have to round up, although there is a way to do it better…

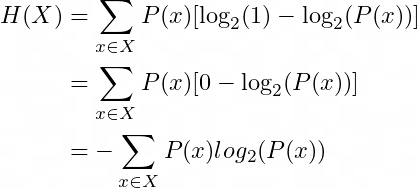
**And the entropy**

We are able to compute the optimal (shortest) length of a binary encoded word using its probability of occurring. However, there is one last step to compute entropy since *“Entropy is the* ***AVERAGE*** *optimal length of an encoded message”,* and that step is to compute the average over all word lengths of our message:



Entropy formula for discrete random variable X. Image by the author.

Developing the log term:



Most common representation of the entropy formula. Image by the author

**Note6**. Usually, you will see the entropy formula using log base 10. In most cases, this doesn’t matter since log\_10(x) = log\_2(x) / log\_2(10) which essentially is multiplying by a constant 1/log\_2(10). This change of base only holds for optimization or comparative methods where the constants are ignored, but it isn’t the same computing the optimal average length of a message using base 2 (each bit of information only takes 2 possible values) as using base 10. Read [this](https://www.khanacademy.org/math/algebra2/x2ec2f6f830c9fb89:logs/x2ec2f6f830c9fb89:change-of-base/a/logarithm-change-of-base-rule-intro) for more information.

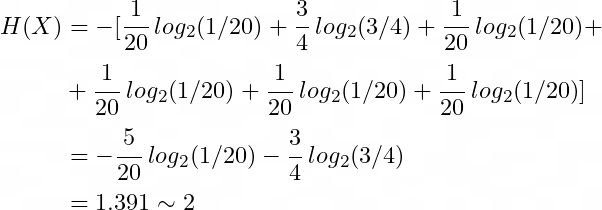
**The final result**

So far I’ve presented 3 different (but related) conceptions of entropy, and derived its expression from scratch using mathematics and visual representations.

However, there is one remaining task. WE HAVE TO SOLVE OUR PROBLEM OF OPTIMALLY ENCODING THE EXPERIMENT RESULTS FOR OUR LABORATORY!

So we just have to plug our experiment’s probability distribution into the entropy formula and compute the terms:

**Note7**. I’m going to use the last probability distribution plotted in the image with the title System 3, but remember that this works the same for any discrete random variable.



Computing the optimal average length for the encoding system of System 3.

Remember that this is an average; thus, some of the experiments might be expressed using 1 bit, others using 2 bits or perhaps 3… Entropy just tells us the average minimum, but it doesn’t reflect how each message should be encoded. **Note that I am using log base 2 to compute entropy since we are using binary encoding** (using log base 10 yields a different result that is weighted by a constant 1/log\_2(10) and doesn’t reflect the real result)

I hope this post has helped you to understand better Entropy. In the next post, I will talk about Cross-entropy and why we use it in Artificial Intelligence models. Finally, I will explain in which problems we apply it and how it can be interpreted.

**References**

**[1]** Visual Information Theory. Christopher Olah. 2015. <https://colah.github.io/posts/2015-09-Visual-Information/>

**[2]** Is ENTROPY Really a “Measure of Disorder”? Physics of Entropy EXPLAINED and MADE EASY. Parth G. 2020. <https://www.youtube.com/watch?v=mg0hueOyoAw>

**[3]** A better description of entropy. Steve Mould. 2016. <https://youtu.be/w2iTCm0xpDc>