towardsdatascience.com

Intuitive Guide to Understanding KL Divergence - Towards Data Science

Thushan Ganegedara

13-16 minutes

Light on Math Machine Learning

I’m starting a new series of blog articles following a beginner friendly approach to understanding some of the challenging concepts in machine learning. To start with, we will start with KL divergence.

Code: Here

The other articles of this series can be found below.

A B C D* E F G* H I J K L* M N O P Q R S T U V W X Y Z

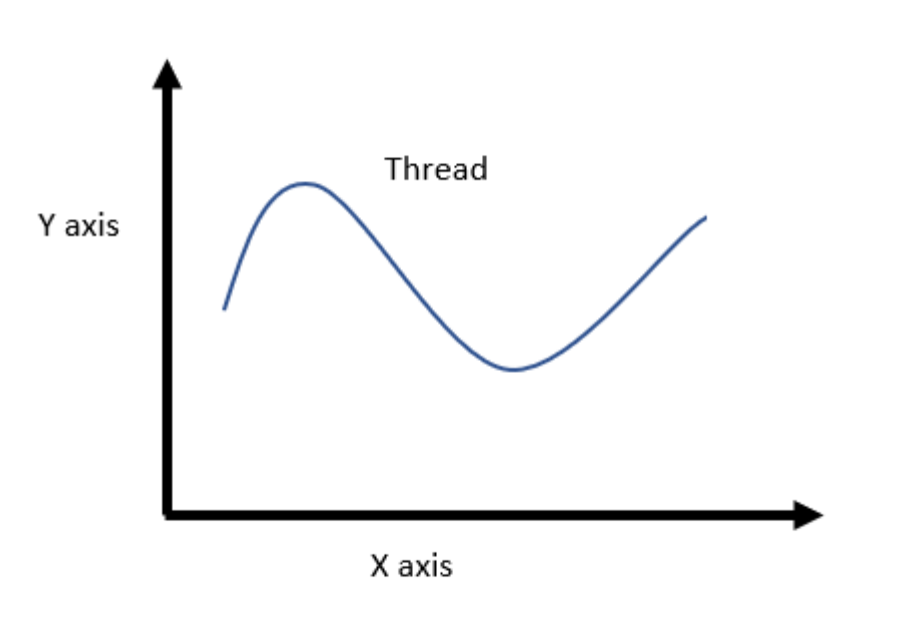
* denotes articles behind the Medium paywall

Concept Grounding

First of all let us build some ground rules. We will define few things we need to know like the back of our hands to understand KL divergence.

What is a Distribution

By distribution we refer to different things such as data distributions or probability distributions. Here we are interested in probability distributions. Imagine you draw two axis (that is, ***X*** and ***Y***) on a paper, I like to imagine a distribution as a thread dropped between the two axis; ***X*** and ***Y***. ***X*** represents different values you are interested in obtaining probabilities for. *Y* represents the probability of observing some value on the ***X*** axis (that is, ***y=p(x)***). I visualize this below.



This is a continuous probability distribution. For example think of axis ***X*** as the height or a human and ***Y*** as the probability of finding a person with that height.

If you want to make this probability distribution discrete, you cut this thread to fixed length pieces and turn the pieces in such a way that they are horizontal. And then create rectangles connecting the edges of each piece of thread and the x-axis. That is a discrete probability distribution.

What is an event?

For a discrete probability distribution, an event is you observing *X* taking some value (e.g. ***X=1***). Let us call ***P(X=1)*** probability of the event ***X=1***. In continuous space you can think of this as a range of values (e.g. ***0.95< X<1.05***). Note that the definition of an event is not restricted to the values it takes on the x-axis. However we can move forward considering only that.

Back to KL divergence

To continue from this point onwards, I will be humbly using the example found in this blog post [1]. It is a great post explaining the KL divergence, but felt some of the intricacies in the explanation can be explained in more detail. All right let's get into it.

Problem we're trying to solve

So the gist of the problem that is being solved in [1] is that, we're a group of scientists visiting the vast outer-space and we discovered some space worms. These space worms have varying number of teeth. Now we need to send this information back to earth. But sending information from space to earth is expensive. So we need need to represent this information with a minimum amount of information. A great way to do this is, instead of recording individual numbers, we draw a plot where ***X*** axis is different numbers of teeth that has been observed (*0,1,2,…, etc.*) and make ***Y*** axis the probability of seeing a worm with *x* many teeth (that is, number of worms with ***x teeth / total number of worms***). We have converted our observations to a distribution.

This distribution is efficient than sending information about individual worms. But we can do better. We can represent this distribution with a known distribution (e.g. uniform, binomial, normal, etc.). For example, if we represent the true distribution with a uniform distribution, we only need to send two pieces of information to recover true data; the uniform probability and the number of worms. But how do we know which distribution explains the true distribution better? Well that's where the KL divergence comes in.

> Intuition: KL divergence is a way of measuring the matching between two distributions (e.g. threads)

So we could use the KL divergence to make sure that we matched the true distribution with some s*imple-to-explain and well-known distribution* well.

Let's change a few things in the example

To be able to check numerical correctness, let us change probability values to more human friendly values (compared to the values used in [1]). We will assume the following. Say we have 100 worms. And we have following types of worms in following amounts.

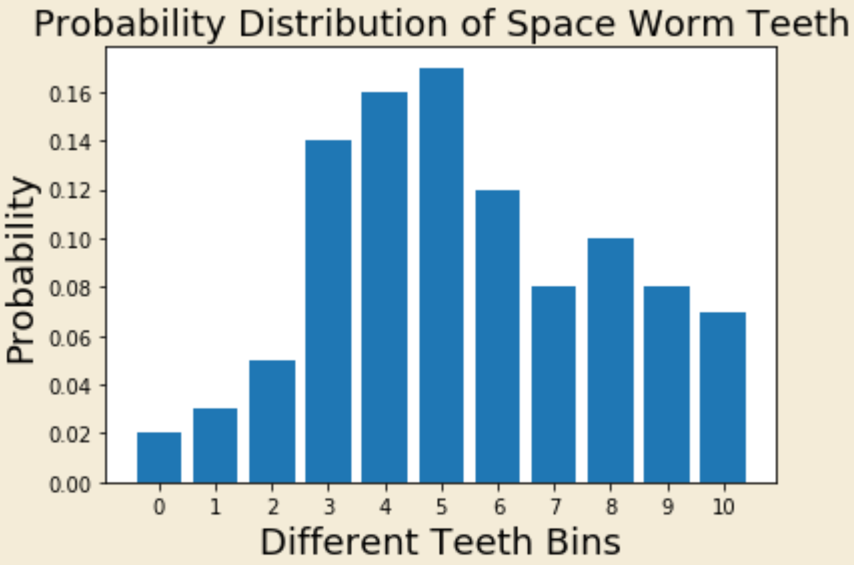
- 0 teeth: 2 (Probability: *p0=0.02*)
- 1 tooth: 3 (Probability: *p1=0.03*)
- 2 teeth: 5 (Probability: *p2=0.05*)

- 3 teeth: 14(Probability: *p3=0.14*)
- 4 teeth: 16 (Probability: *p4=0.16*)
- 5 teeth: 15 (Probability: *p5=0.15*)
- 6 teeth: 12 (Probability: *p6=0.12*)
- 7 teeth: 8 (Probability: *p7=0.08*)
- 8 teeth: 10 (Probability: *p8=0.1*)
- 9 teeth: 8 (Probability: *p9=0.08*)
- 10 teeth: 7 (Probability: *p10=0.07*)

Quick sanity check! Let’s ensure that the values add up to 100 and probability add up to 1.0.

Total worms=2+3+5+14+16+15+12+8+10+8+7 = 100
Total probability=0.02+0.03+0.05+0.14+0.16+0.15+ ….
0.12+0.08+0.1+0.08+0.07=1.0

Here’s what it looks visually.

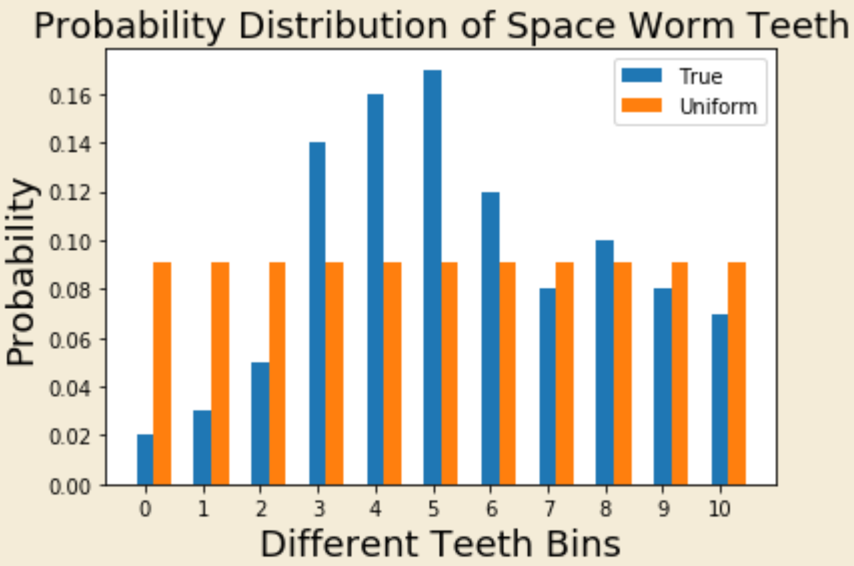


First try: Model this with a uniform distribution

Now that out of the way, let us first try to model this distribution with a uniform distribution. A uniform distribution has only a single parameter; the uniform probability; the probability of a given event happening.

p_uniform=1/total events=1/11 = 0.0909

This is what the uniform distribution and the true distribution side-by-side looks like.



Let us keep this result aside and we will model the true distribution with another type of distributions.

Second try: Model this with a binomial distribution

You are probably familiar with the binomial probability through calculating probabilities of a coin landing on it's head. We can extend the same concept to our problem. For a coin you have two possible outputs and assuming the probability of the coin landing on its head is ***p*** and you run this experiment for ***n*** trials, the probability getting ***k*** successes is given by,

$$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

Breaking down the equation

Let's take a side trip and understand each term in the binomial distribution and see if they make sense. The first term is ***p^k***.

We want to get ***k*** successes, where the probability of a single success is ***p***. Then the probability of getting ***k*** successes is ***p^k***. Remember that we're running the experiment for ***n*** trials. Therefore, there's going to be ***n-k*** failed trials, with a failure probability of ***(1-p)***. So the probability of getting ***k*** successes is the joint probability of ***p^k (1-p)^{n-k}***. Our work doesn't end here. There are different permutations the ***k*** trials can take place within the *n* trials. The number of different permutations *k* elements to be arranged within ***n*** spaces is given by,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Multiplying all these together gives us the binomial probability of ***k*** successes.

Mean and variance of the binomial distribution

We can also define a mean and a variance for a binomial distribution. These are given by,

mean = np
variance = np(1-p)

What does the mean reflect? Mean is the expected (average) number of successes you get if you run *n* trials. If each trial has a success probability of *p* it make sense to say you will get ***np*** successes if you run ***n*** trials. Next what does the variance represent. It represents how much the true number of success trials to deviate from the mean value. To understand the variance, let us assume ***n=1***. Then the equation is, ***variance=p(1-p)***. You have the highest variance when ***p=0.5*** (when it is equally likely to get heads and tail) and lowest when ***p=1*** or ***p=0*** (when for sure you're getting head/tail).

Back to modeling

Now with a solid understanding about the binomial distribution, let us spiral back to the problem at our hands. Let us first calculate the expected number of teeth for the worms. It would be,

$$\begin{aligned} &0 \times p_0 + 1 \times p_1 + 2 \times p_2 + \ldots + 10 \times p_{10} \\ &= 0 \times 0.02 + 1 \times 0.03 + 2 \times 0.05 + \ldots + 10 \times 0.07 \\ &= 5.44 \end{aligned}$$

With mean known, we can calculate ***p*** where,

mean = np

5.44 = 10p

p = 0.544

Note than ***n*** is the maximum number of teeth observed from the population of worms. You might ask why we did not choose ***n*** to be the total number of worms (that is ***100***) or total number of events (that is ***11***). We will soon see the reason. With that, we can define probabilities of any number of teeth as follows.

> Given that teeth can take values up to 10, what is the probability of seeing k teeth (where seeing a tooth is a success trial).

From the perspective of the coin flip, this is like asking,

> Given that I have 10 flips, what is the probability of observing k heads.

Formally, we calculate the probability ***pk^{bi}*** for all different values of ***k***. Here ***k*** becomes the number of teeth we would like to observe. And ***pk^{bi}*** is the binomial probabilities for the ***k th*** bin of teeth (that is, 0 teeth, 1 tooth, etc.). So when we calculate them as follows,

p0^{bi} = (10!/(0!10!)) 0.544^0 (1–0.544)^{10} = 0.0004

$p1^{bi} = (10!/(1!9!))\ 0.544^1\ (1–0.544)^9 = 0.0046$

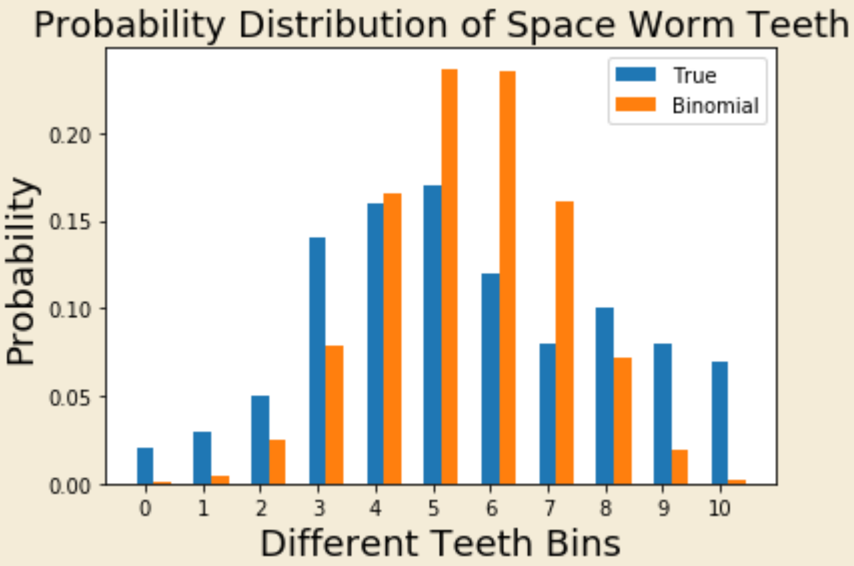
$p2^{bi} = (10!/(2!8!))\ 0.544^2\ (1–0.544)^8 = 0.0249$

…

$p9^{bi} = (10!/(9!1!))\ 0.544^9\ (1–0.544)^1 = 0.0190$

$p10^{bi} = (10!/(10!0!))\ 0.544^{10}\ (1–0.544)^0 = 0.0023$

This is what a comparison between the true distribution and the binomial distribution looks like.

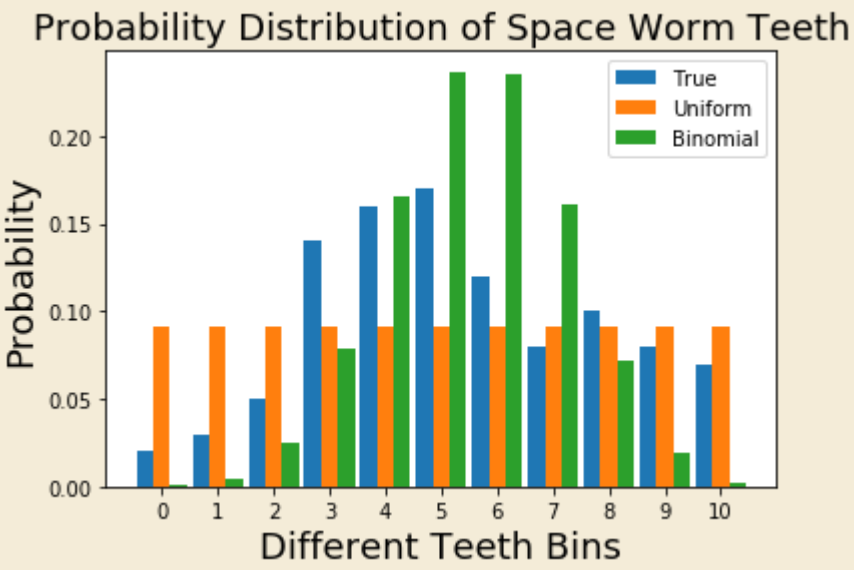


Let’s summarize what we have

Okey, turn back and reflect on what we did so far. First we understood the problem we want to solve. The problem is to send statistics of teeth of a certain type of space worms across the space with minimal effort. For that we thought of representing the true statistics of worms with some known distribution, so we can just send the parameter of that distribution instead of true statistics. We looked at two types of distributions and came up with the following statistics.

* Uniform distribution — with probability of ***0.0909***
* Binomial distribution — with ***n=10***, ***p=0.544*** and ***k*** taking different values between 0 to 10

Now let’s visualize everything in one place



How do we quantitatively decide which ones the best?

Now with all these fancy calculations, we need a way to measure the matching between each approximated distribution and the true distribution. This is important, so that, when we send the information across, we can have a peace of mind without worrying about the question "did I choose correctly?" for the rest of our lives.

This is where the KL divergence comes in. KL divergence is formally defined as follows.

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)})$$

Here ***q(x)*** is the approximation and ***p(x)*** is the true distribution we're interested in matching ***q(x)*** to. Intuitively this measures the how much a given arbitrary distribution is away from the true distribution. If two distributions perfectly match, ***D_{KL} (p||q) = 0*** otherwise it can take values between ***0*** and ∞. Lower the KL divergence value, the better we have matched the true distribution with our approximation.

Intuitive breakdown of the KL divergence

Let's look at the KL divergence piece by piece. First take the ***log(p(x_i)/q(x_i))*** component. What happens if ***q(x_i)*** is higher than ***p(x_i)***? Then this component will produce a negative value (because log of less than 1 values are negative). On the other hand if ***q(x_i)*** is always smaller than ***p(x_i)*** this component will produce positive values. This will be zero only if ***p(x_i)=q(x_i)***. Then to make this an expected value, you weight the log component with ***p(x_i)***. This means that, matching areas where ***p(x_i)*** has higher probability is more important than matching areas with low ***p(x_i)*** probability.

Intuitively it makes sense to give priority to correctly match the truly highly probable events in the approximation. Mathematically, this allows you to automatically ignore the areas of the distribution that falls outside of the support (support is the full length on the x axis used by a distribution) of the true distribution. Additionally this avoid calculating ***log(0)*** that will come up if you try to compute the log component for any area that falls outside of the support of the true distribution.

Computing KL divergence

Let us now compute the KL divergence for each of the approximate distributions we came up with. First let's take the uniform distribution.

$$D_{KL}(True||Uniform) = 0.02(0.02/0.0909) + 0.03(0.03/0.0909) + \ldots$$

$$+0.08(0.08/0.0909) + 0.07(0.07/0.0909)$$

$$D_{KL}(True||Uniform) = 0.136$$

Now for the binomial distribution we get,

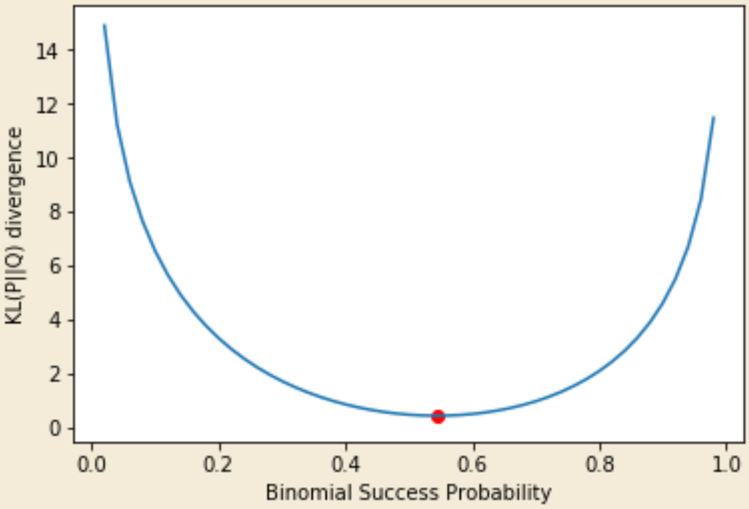
$$D_{KL}(True||Binomial) = 0.02 \times log(0.02/0.0004) + 0.03 \times log(0.03/0.0046) + \ldots$$

$$+0.08 \times log(0.08/0.0190) + 0.07 \times log(0.07/0.0023)$$

$$D_{KL}(True||Binomial) = 0.427$$

KL Divergence with respect to Binomial Mean

Let's just play around with the KL divergence now. First we will see how the KL divergence changes when the success probability of the binomial distribution changes. Unfortunately we cannot do the same with the uniform distribution because we cannot change the probability as ***n*** is fixed.



You can see that as we are moving away from our choice (red dot), the KL divergence rapidly increases. In fact, if you print some of the KL divergence values small **Δ** amount away from our choice, you will see that our choice of the success probability gives the minimum KL divergence.

Now we arrive to the end of our discussion about KL divergence.

Conclusion

Now we have some solid results, though the uniform distribution appears to be simple and very uninformative where the binomial distribution carries more subtlety, the uniform distribution matches the true distribution better than the binomial distribution. To be honest, this result actually took me by surprise. Because I expected the binomial to model the true distribution better. Therefore, this teaches us the important less of why we should not trust our instincts alone!

Code: Here

Fun with KL divergence

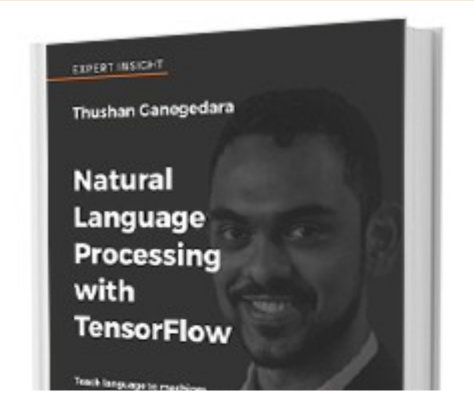
You can have more fun around playing with the KL divergence to understand KL divergence better. You can read more about this in my blog post.

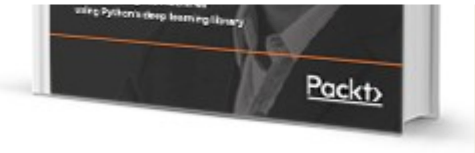
Reference

[1] https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

Note: Please go and checkout my website as I post more machine learning stuff there as well.

Small note: I'm pleased to announce that, my book on natural language processing with TensorFlow has been released and is up for grabs! The book is ideal for beginner/intermediate level readers seeking a practical perspective of modern deep learning based solutions. The book is accompanied with exercises guiding the reader to implement a variety of NLP applications. You can find it in the Packt website or Amazon.




using Python's deep learning library
Packt>