Learning KL Divergence: A Python Tutorial with Examples

Authored by

Mohammed looti

November 1, 2025

The **Kullback-Leibler (KL) divergence** stands as a foundational concept within the fields of statistics and Information theory. Its primary function is to provide a quantitative measure of the difference between two competing probability distributions.

In the realm of machine learning, especially in tasks such as model optimization and variational inference, KL divergence is indispensable. It allows practitioners to assess the efficacy of a proposed distribution (often labeled Q) when it is used to approximate a true or reference distribution (P). Fundamentally, the resulting value quantifies the amount of information lost when distribution Q is substituted for the true distribution P.

When working with two discrete probability distributions, P and Q, the divergence of P from Q is formally denoted using the notation **KL(P || Q)**. Understanding this precise notation is vital, as it highlights the directional nature of the comparison, which we will explore in detail later.

The Mathematical Foundation of KL Divergence

The **Kullback-Leibler divergence** is calculated using a rigorous mathematical formula derived directly from information theoretic principles established by Solomon Kullback and Richard Leibler. For discrete probability distributions, the calculation involves summing across all possible events the product of the true probability P(x) and the logarithm of the ratio between P(x) and the approximating probability Q(x).

This relationship is formally expressed using the following summation formula:

$$KL(P \,||\, Q) = \Sigma P(x) \, ln(P(x) / Q(x))$$

A crucial property of the **KL divergence** is that its value is inherently non-negative. A result of exactly zero occurs only if the two distributions, P and Q, are statistically identical. Any result greater than zero signifies that information is lost when Q is used as an approximation for P, with larger values indicating greater divergence or disparity between the two distributions.

This formula is pivotal in advanced statistical methodologies, including Bayesian inference and various model optimization routines. The objective in these applications is often to minimize this divergence, driving the approximating distribution Q as close as possible to the reference distribution P.

Why KL Divergence is an Asymmetric Measure

Although the **KL divergence** is frequently referred to as a "distance" between distributions, it is mathematically inaccurate to categorize it as a true distance metric. A valid mathematical distance metric must satisfy three strict criteria: non-negativity, identity of indiscernibles (distance is zero only if the points are identical), and crucially, symmetry (the distance from A to B must equal the

distance from B to A).

The KL divergence successfully satisfies the first two properties, but it fundamentally fails the third, classifying it as an **asymmetric measure**. This critical distinction means that calculating **KL(P || Q)** will almost always yield a different result than calculating **KL(Q || P)**.

The directionality inherent in the calculation carries significant interpretive meaning:

KL(P || Q): This measures the expected excess surprise or loss incurred when using Q as the predictive model, given that P is the true underlying distribution.
KL(Q || P): Conversely, this measures the expected excess surprise when using P as the predictive model, assuming Q is the true underlying distribution.

Acknowledging this asymmetry is essential for practical applications, particularly in advanced techniques like variational inference, where the choice between forward KL (P || Q) and reverse KL (Q || P) significantly impacts the resulting model bias and the nature of the approximation achieved.

Implementing KL Divergence using Python and SciPy

Calculating the **KL divergence** using Python is highly efficient, largely thanks to powerful scientific computing libraries. The SciPy library, a cornerstone of numerical computation in Python, offers a specialized function tailored for this calculation within its `scipy.special` module.

The specific function utilized is `rel_entr`, which calculates the relative entropy--another name for KL divergence. This function computes the element-wise contribution of each event to the total divergence according to the formula. To obtain the final, scalar KL divergence value, it is necessary to sum these element-wise results.

Before any calculation can be deemed valid, a strict mathematical prerequisite must be met: both probability distributions (P and Q) must be properly normalized. This means that the sum of all probabilities within each distribution must exactly equal one (1.0). If this requirement is not satisfied, the calculation of the **KL divergence** will be mathematically invalid and the results meaningless.

Practical Example: Calculating KL Divergence in Python

To demonstrate the computational approach, let us define two discrete probability distributions, P and Q, as Python lists. These distributions represent the probabilities of eight distinct and mutually exclusive outcomes. It is important to confirm that the probabilities for both distributions sum precisely to 1.0, validating them as legitimate probability mass functions.

#define two probability distributions
P =
Q =

We proceed by using the `rel_entr` function from the SciPy library. The following code snippet calculates the divergence of P from Q, **KL(P || Q)**, by first computing the relative entropy terms and then summing them to yield the final divergence value:

```
from scipy.special import rel_entr

#calculate KL(P || Q)
sum(rel_entr(P, Q))
```

0.589885181619163

The resulting **KL divergence**, **KL(P || Q)**, is approximately **0.590**. Since this value is positive and non-zero, it confirms that the approximating distribution Q is not a perfect representation of the true distribution P, and there is a quantifiable amount of information lost when Q is used in its place.

Empirical Demonstration of Asymmetry: Calculating KL(Q || P)

To definitively demonstrate that the **Kullback-Leibler divergence** is not a symmetric metric, we must swap the order of the input distributions and calculate **KL(Q || P)**. If the measure is asymmetric, this reversed calculation must yield a different numerical result than the previous one.

The calculation methodology remains identical, but the arguments passed to the `rel_entr` function are reversed, placing Q first and P second:

```
from scipy.special import rel_entr

#calculate KL(Q || P)
sum(rel_entr(Q, P))
```

0.497549319448034

The **KL divergence** of distribution Q from distribution P, **KL(Q || P)**, is calculated to be approximately **0.498**. Since 0.590 is clearly not equal to 0.498, this example successfully validates the fundamental **asymmetry** of the measure. We interpret this difference by recognizing that the information loss associated with approximating P using Q is not the same as the loss incurred when approximating Q using P.

Interpreting the Units of Divergence: Nats vs. Bits

The numeric value obtained from any **KL divergence** calculation carries a unit of measure determined by the base of the logarithm used in the underlying formula. In the Python example utilizing the SciPy function `rel_entr`, the library employs the natural logarithm (log base *e*).

Consequently, the units for the divergence calculation are known as **nats**, an abbreviation for *natural unit of information*. Therefore, we should accurately state that the result KL(P || Q) is **0.590 nats**.

It is important to note that alternative formulations for calculating **KL divergence** often substitute the natural logarithm with log base-2. When log base-2 is used, the divergence is expressed in terms of **bits** (also known as shannons) instead of nats. While the choice of unit does not alter the underlying statistical relationship between P and Q, it does change the scale of the numeric result. Maintaining strict consistency in unit usage is essential when comparing results across different models or research studies.

Further Exploration in Information Theory and Statistics

To deepen your expertise in statistical comparisons and their practical implementation within Python environments, consider exploring these closely related advanced topics:

Calculating Statistical Entropy and Cross-Entropy for probability distributions.
Advanced statistical testing and distribution analysis using the extensive toolset provided by the SciPy library.
A detailed comparative analysis of the **Kullback-Leibler divergence** against the symmetric Jensen-Shannon Divergence.