Information Theory

Application in Data Science

By Chiranjit Pathak

Information Theory & it's measurements

Information theory studies the transmission, processing, extraction, and utilization of information. Abstractly, information can be thought of as the resolution of uncertainty.
A cornerstone of information theory is the idea of quantifying how much information there is in a message. More generally, this can be used to quantify the information in an event and a random variable, called entropy, and is calculated using probability

Information is only useful when it can be stored and/or communicated. In a digital form, information is stored in 'bits', or a series of numbers that can either be 0 or 1. The letters in keyboard are stores in a 'byte', which is 8 bits, which allows for 2^8 =256 combinations.

The mathematician Claude Shannon had the insight that the more predictable some information is, the less space is required to store it. Shannon had a mathematical formula for the **'entropy'** of a probability distribution, which outputs the minimum number of bits required, on average, to store its outcomes.

Quantifying information is the foundation of the field of information theory.
The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

We can calculate the amount of information there is in an event using the probability of the event. This is called "*Shannon information*," "*self-information*," or simply the "*information*," and can be calculated for a discrete event *x* as follows:

- information(x) = -log(p(x))

Where *log()* is the base-2 logarithm and *p(x)* is the probability of the event *x*.

The choice of the base-2 logarithm means that the units of the information measure is in bits (binary digits). This can be directly interpreted in the information processing sense as the number of bits required to represent the event.

The calculation of information is often written as *h()*; for example:

- h(x) = -log(p(x))

The negative sign ensures that the result is always positive or zero.

Information will be zero when the probability of an event is 1.0 or a certainty, e.g. there is no surprise.

Entropy can be calculated for a random variable X with k in K discrete states as follows:
H(X) = -sum(each k in K p(k) * log(p(k)))
That is the negative of the sum of the probability of each event multiplied by the log of the probability of each event.

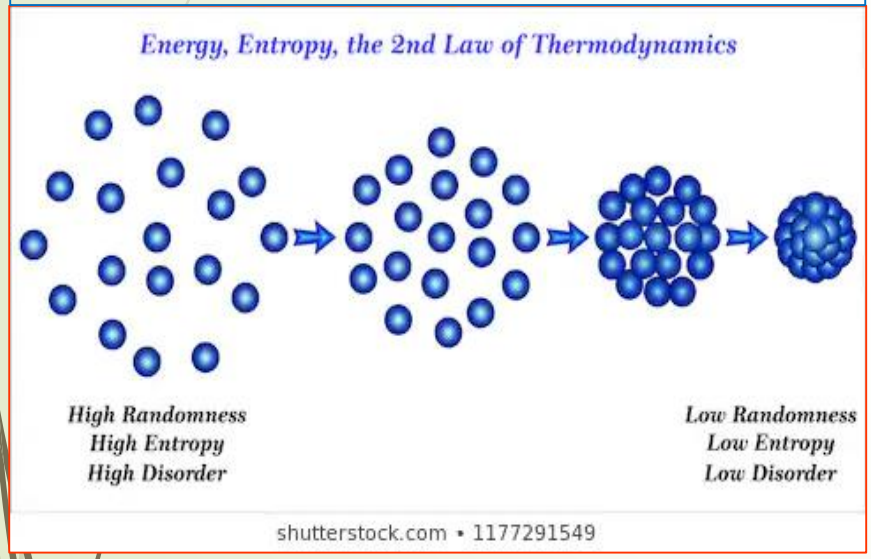
Calculating the entropy for a random variable provides the basis for other measures such as **mutual information** or **information gain**.

Analogy with Physics & its Application in DS

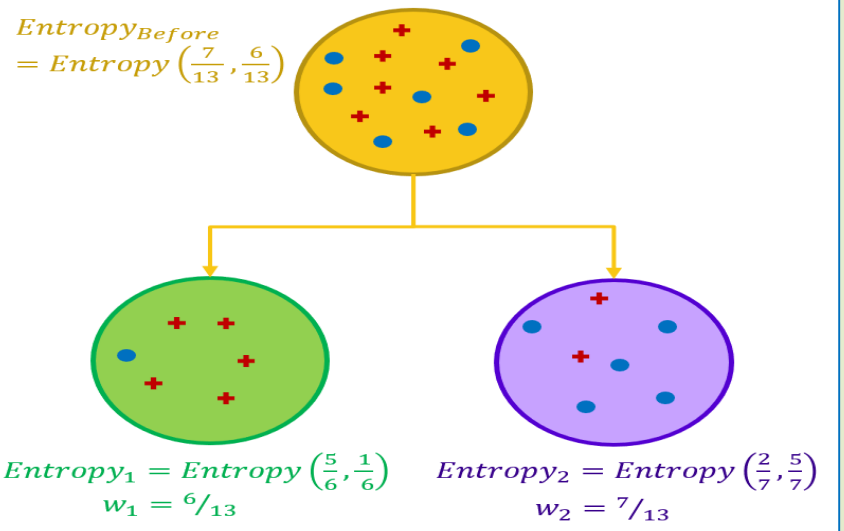
3

The macroscopic state of a system is characterized by a distribution on the microstates.
The entropy of this distribution is given by the Gibbs entropy formula.
For a classical system (i.e., a collection of classical particles) with a discrete set of microstates, if p{i} is the probability that it occurs during the system's fluctuations, then the entropy of the system is,
$S = -k_B \sum_i p_i \ln p_i$
The quantity k{B} is a physical constant known as Boltzmann's constant.

Analogy with Thermodynamics: Entropy high (Gas) to Zero(Solid)



Application in DS: Tree Based Algorithm Target Entropy of Leaf node is Zero



The particles inside pure solid are tightly bind and the degree of disorderness is almost zero making it is a pure & homogenous in nature and hence entropy is zero.

Applications:
Shannon's work found uses in data storage, spaceship communication, and even communication over the internet. 'KL divergence' is an idea derived from Shannon's work, that is frequently used in data science. It tells how good one distribution is at estimating another by comparing their entropies.

Parameters Derived to use in ML

4

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making.

Impurity - Entropy & Gini

There are three commonly used impurity measures used in binary decision trees: **Entropy**, **Gini index**, and **Classification Error**.

Entropy (a way to measure impurity):

$$Entropy = -\sum_j p_j \log_2 p_j$$

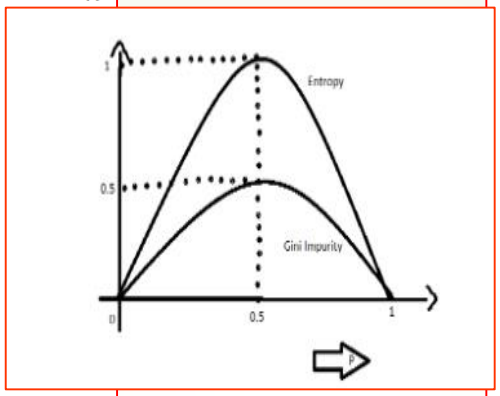
Gini index (a criterion to minimize the probability of misclassification):

$$Gini = 1 - \sum_j p_j^2$$

Classification Error:

$$ClassificationError = 1 - \max p_j$$

where p_j is the probability of class j.



1. Entropy of a group in which all examples belong to the same class:

$$entropy = -1 \log_2 1 = 0$$

This is not a good set for training.

2. entropy of a group with 50% in either class:

$$entropy = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

This is a good set for training.

So, basically, the entropy attempts to maximize the mutual information (by constructing a equal probability node) in the decision tree.

Similar to entropy, the **Gini index** is maximal if the classes are perfectly mixed, for example, in a binary class:

$$Gini = 1 - (p_1^2 + p_2^2) = 1 - (0.5^2 + 0.5^2) = 0.5$$

Using a decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG).
Basically, using IG, we want to determine which attribute in a given set of training feature vectors is most useful.
In other words, IG tells us how important a given attribute of the feature vectors is.

Applications:

We **repeat this splitting procedure** at each child node down to the empty leaves. This means that the samples at each node all belong to the same class.

However, this can result in a very deep tree with many nodes, which can easily **lead to overfitting**. Thus, we typically want to **prune** the tree by setting a limit for the **maximum depth of the tree**.

The **Information Gain (IG)** can be defined as follows:

$$IG(D_p) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

where I could be **entropy**, **Gini index**, or **classification error**, D_p, D_{left}, and D_{right} are the dataset of the parent, left and right child node.

The internal working of both methods is very similar. But if we compare both the methods then Gini Impurity is more efficient than entropy in terms of computing power.
The range of **Entropy lies in between 0 to 1** and the range of **Gini Impurity lies in between 0 to 0.5**.
Hence we can conclude that Gini Impurity is better as compared to entropy for selecting the best features.

References:
1. https://machinelearningmastery.com/what-is-information-entropy/
2. https://towardsdatascience.com/information-entropy-c037a90de58f
3. https://en.wikipedia.org/wiki/Entropy_(statistical_thermodynamics)#Gibbs_entropy_formula
4. https://www.bogotobogo.com/python/scikit-learn/scikt_machine_learning_Decision_Tree_Learning_Informatioin_Gain_IG_Impurity_Entropy_Gini_Classification_Error.php
5. https://www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml/

Thanks for reading!

Lets collaborate and happy to receive any feedback/suggestion/comment at…….

pathak.chiranjit@gmail.com