Entropy: a basic introduction

1) Entropy = measure of randomness
2) Entropy = measure of compressibility

More random = Less compressible

High entropy = high randomness/low compressibility
Low entropy = low randomness/high compressibility

Entropy is a key notion applied in information retrieval and data compression in general.

Entropy application

Entropy enables one to

compute the compressibility of data

without actually needed to compress the data first!

We will illustrate this with a well-known file compression method: the Huffman algorithm

Huffman encoding example

We compute the Huffman code and measure the compression of the file.

This is compared to the “entropy”, a measure of file compressibility obtained from the file (without the need to actually compress)

Probabilities and randomness

6-sided fair dice

pi = [Probability of outcome = i] = 1/6
Where i is any number from 1 to 6

6-sided biased dice

p6 = 3/12 = ¼ (6 is more likely)
p1 = 1/12 (1 is less likely, piece of led in the "dot")
p2 = p3 = p4 = p5 = 2/12 = 1/6

Sum of probabilities of all possible outcomes is 1

$$p1 + p\,2 + p3 + p4 + p5 + p6 = 1$$

Probabilities (general case)

For the general case, instead of 6 outcomes (dice) we allow n outcomes.

For instance, consider a file with 100,000 characters. Say the character "a" occurs 45,000 times in this file. What is the probability of encountering "a" if we pick a character in the file at random?

Answer: 45/100 = 0.45 i.e. nearly half of the file consists of "a"-s

Say there are n characters in the file.
Each character will have a probability pi.
Sum of the probabilities ($p_1 + \ldots + p_n$) = 100/100 = 1

Entropy

Given a "probability distribution", i.e. given probabilities, $p_1, \ldots, p_n$, with sum 1, then we define the entropy H of this distribution as:

$$H(p_1,\ldots,p_n) = -p_1\log(p_1) - p_2\log(p_2) - \ldots - p_n\log(p_n)$$

Note: log has base 2 in this notation and
$\log_2(k) = \ln(k)/\ln(2)$
(where "ln" is "log in base e")

Exercise: compute the entropy for

a) the probability distribution of the fair dice
b) the probability distribution of the biased dice

Comment on logs

– plog(p) = plog(1/p)

$$\log(1/p) = \log(1) - \log(p)$$
$$= 0 - \log(p)$$
$$= -\log(p)$$

IMPORTANT: plog(1/p) measures a very intuitive concept

* p is the probability of an event
* 1/p is the number of times the event occurs
* log(k) measures how many bits are needed to represent the outcomes

We check this on the fair dice

Fair dice example

p = 1/6
p = probability of an outcome = 1/6

1/p = 1/(1/6) = 6
1/p = number of outcomes = 6

log(1/p) = log(6) = 2.59
log(1/p) = log(number of outcomes)
= "number" of bits needed to represent the 1/p = 6 outcomes

Rounding up

Note: in general "number" of bits, i.e. log(1/p), is not a positive integer. E.g. 2.59. In practice we can take the smallest integer greater than or equal to log(1/p), which is 3

Note that 3 bits suffice to represent the 6 outcomes:

Binary numbers of length 3, there are 8 of them, so pick six of them to represent the outcomes of the dice, e.g.

000, 001, 010, 011, 100 and 101

Comment: in the following we don't round up (we will see why).

Nice Entropy interpretation

$H(p_1,\ldots,p_n)$ = average encoding length

We have shown that $-p\log(p) = p\log(1/p)$

So our entropy can be written as:

$H(p_1,\ldots,p_n) = p_1\log(1/p1) + p_2\log(1/p2) + \ldots + p_n\log(1/pn)$

Where

$p_i\log(1/p_i)$ =

probability of occurrence x encoding length

Thus: $H(p_1,\ldots,p_n)$ = average encoding length

Binary represenation

To encode n distinct numbers in binary notation
we need to use binary numbers of length log(n)
Note that from here on “log” will be the logarithm in base 2 since we are interested in binary compression only.

To encode 6 numbers, we need to use binary numbers of length log(6) (in fact, we need to take the nearest integer above this value, i.e. 3). Binary numbers of length 2 will not suffice (there are only 4 which is not suitable to encode 6 numbers). We keep matters as an approximation and talk about binary numbers of “length” log(6), even though this is not an integer value.

Binary number length to encode 8 numbers is log(8) = 3

Exercise: solution

a) Fair dice: p1 = p2 = … = p6 = 1/6
So H(p1,…,p6) = -1/6log(1/6) X 6
= - log(1/6)
= --log(6)
= log(6) = 2.59

Interpretation: Entropy measures the amount of randomness. In the case of a fair dice, the randomness is "maximum". All 6 outcomes are equally likely. This means that to represent the outcomes we will "roughly" need log(6) = 2.59 bits to represent them in binary form (the form compression will take).

Solution continued

b) The entropy for the biased dice is:

$-1/4\log(1/4) - 3/20\log(3/20) \times 5 =$

$1/4\log(4) + 3/20\log(20/3) \times 5 =$

$¼ \times 2 + 3/4 \log(20/3) =$

$½ + 3/4\log(20/3) =$

$0.5 + 2.055 =$

2.555 (Lower than our previous result!)

Exercise continued

Try the same for an 8-sided dice
(dungeons and dragons dice) which is

a) Fair
b) Totally biased, with prob(8) =1
and thus prob(1) = … = prob(7) = 0

Answers:

a) Entropy is log(8) = 3, we need 3 bits to represent the 8 outcomes (maximum randomness)
b) Entropy is 1log(1) = 0, we need a bit of length 0 to represent the outcome. Justify! (Note: bit of length 1 has 2 values. Bit of length 0 has ? Values).

Compression

Revisit previous example of 8-sided dice

Compression for outcomes of fair dice:

No compression (we still need 8 values to encode)
(Maximum randomness)
(outcome of entropy/total number of values) = 8/8 = 1

Compression for outcomes of biased dice:

Total compression (we only need 1 bit to encode)
("No" randomness)
(outcome of entropy/total number of values) = 0/8 = 0

Exercise: Huffman code

Consider a file with the following properties:

Characters in file: a,b,c,d,e and f

Number of characters: 100,000

Frequencies of characters
(in multiples of 1,000): freq(a) = 45, freq(b) = 13,
freq(c) = 12, freq(d) = 16,
freq(e) = 9, freq(f) = 5

So “a” occurs 45,000 times and similar for the others

Exercise continued

a) Compute the Huffman encoding
b) Compute the cost of the encoding
c) Compute the average length of the encoding
d) Express the probability of encountering a character in the file (do it for each character)
e) Compute the Entropy
f) Compare the Entropy to the compression percentage

What is your conclusion?

Solution

We assume familiarity with the Huffman code Algorithm.

Answer:
a) (prefix) codes for characters:
a: 0, b: 101, c:100, d: 111, e: 1101, f: 1100

b) Cost of encoding = number of bits in encoding =
45 x 1 + 13 x 3 + 12 x 3 + 16 x 3 + 9 x 4 + 5 x 4 =
224,000 bits

c) 224,000/100,000 = 2.24 average encoding length

d) Prob(char = a) = 45/100, …, Prob(char = f) = 5/100
Check: sum of probabilities = 100/100 = 1

Solution (continued)

e) Entropy =

H(45/100, 13/100, 12/100, 16/100, 9/100, 5/100) =

- 45/100log(45/100) - 13/100log(13/100)
- 12/100log(12/100) - 16/100log(16/100)
- 9/100log(9/100) - 5/100log(5/100) = 2.23

f) Conclusion: Entropy is an excellent prediction of average binary encoding length (some minor round-off errors). It predicted the average code length to be 2.23, very close to 2.24. It also predicts total size of compressed file: 2.23 x 100,000 = 223,000 which is very close to actual compressed size: 224,000