

ENTROPIC QUESTIONING

NACHUM

1. INTRODUCTION

Goal. Pick the question that contributes most to finding a suitable product.

Idea. Use an information-theoretic measure.

Basics. Entropy (a non-negative real number) measures the amount of uncertainty inherent in a distribution: high entropy corresponds to large uncertainty; zero entropy corresponds to perfect knowledge.

Let $P = \{p_1, \dots, p_n\}$ be a discrete probability distribution (set of probabilities), corresponding to events $\{c_1, \dots, c_n\}$. The *entropy* of P is

$$H_P = - \sum_i p_i \log p_i$$

By definition, set $0 \log 0 = 0$.

We will use binary logarithms, $\lg (= \log_2)$, instead of the usual natural logarithms, \ln , the difference being only a multiplicative constant.

Example. For $n = 4$, we have:

P	H	
$(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$	2	uniform
$(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, 0)$	1.5	one eliminated
$(\frac{1}{2}, \frac{1}{2}, 0, 0)$	1	two eliminated
$(\frac{3}{4}, \frac{1}{4}, 0, 0)$	0.8	skewed
$(0, 1, 0, 0)$	0	certainty

2. METHOD

Theory. Let events c_i be successes (item i is the right one). The (binary) entropy of the system is

$$H = - \sum_i \Pr(c_i) \lg \Pr(c_i)$$

After asking a question (attribute) A and getting an answer $j \in A$, the entropy becomes

$$H_j = - \sum_i \Pr(c_i|j) \lg \Pr(c_i|j)$$

Date: November 2005. These ideas were first conceived in Spring 2001.

where $\Pr(c_i|j)$ is the posterior probability of c_i , given answer j . Thus, the expected updated entropy after a question with answers $j \in A$ is

$$\bar{H}_A = - \sum_j \Pr(j) H_j$$

where $\Pr(j)$ is the prior probability that answer j will be given.

Thus, \bar{H}_A measures the expected uncertainty after receiving an answer from A . The lower \bar{H} is, the more information obtained by asking the question.

Example. Consider questions with the following answer distributions:

A	Y	N	B	Y	N	C	Y	N
a	0	1	a	$\frac{1}{2}$	$\frac{1}{2}$	a	0	1
b	0	1	b	$\frac{1}{2}$	$\frac{1}{2}$	d	$\frac{1}{2}$	$\frac{1}{2}$
c	1	0	c	$\frac{1}{2}$	$\frac{1}{2}$	c	1	0
d	1	0	d	$\frac{1}{2}$	$\frac{1}{2}$	d	$\frac{1}{4}$	$\frac{3}{4}$

Suppose that initially four items $\{a, b, c, d\}$ are equiprobable, with entropy 2. Question A gives a system with entropy 1, for either answer. Question B is meaningless, and leaves the entropy as is, regardless of the answer. Question C reduces the entropy to 1.125 if the answer is Y , and to 1.328 for N , for a weighted average of 1.239.

So, clearly A is best. Having asked A and gotten an answer, both A and B are worthless.

Note. When there are many answers j , each H_j will be relatively small, in which case \bar{H} will also be small—assuming that the question is meaningful.

3. APPLICATION

Parameters.

S	Set of (relevant) items
$n = S $	Number of (relevant) items
$p_i = \Pr(c_i)$	A priori likelihood (normalized rank) that item $i \in S$ will be chosen
$q_{ij} = \Pr(j c_i)$	Probability of choice (answer) j for desired item i
$r_j = \Pr(j)$	Probability of choice j
$= \sum_i p_i q_{ij}$	
$p_i^j = \Pr(c_i j)$	Posterior probability for item i , assuming choice j
$= p_i q_{ij} / r_j$	
$H_j = - \sum_i p_i^j \lg p_i^j$	Entropy after choosing j
$\bar{H} = \sum_j r_j H_j$	Expected entropy (after answering given question)

Method. Given the set S and probabilities p_i and q_{ij} , choose the question with smallest \bar{H} value. Update each p_i to p_i^j , where j is the answer obtained.

Additionally, one may want to ask no question when none make a significant improvement in entropy.

Background. The use of entropy calculations to design decision trees was suggested by Ross Quinlan in his famous ID3 algorithm [3].

Assumptions.

- All products in S with non-zero probability appear under at least one choice (“other”, if necessary). If not, then some large “cost” should be assigned them.
- Each leaf corresponds to a single product or to any of a number of indistinguishable products (all of which will satisfy the customer equally).

Facts.

$$\sum_i p_i = \sum_j q_{ij} = \sum_j r_j = 1$$

Simplifications.

- (1) The above entropy formula for a particular question is

$$\begin{aligned} \bar{H} &= \sum_j r_j H_j \\ &= \sum_j r_j \left(- \sum_i \frac{p_i q_{ij}}{r_j} \lg \frac{p_i q_{ij}}{r_j} \right) \\ &= - \sum_j \left(\sum_i p_i q_{ij} \lg \frac{p_i q_{ij}}{\sum_i p_i q_{ij}} \right) \\ &= \sum_j \sum_i p_i q_{ij} \lg \left(\sum_i p_i q_{ij} \right) - \sum_j \sum_i p_i q_{ij} \lg p_i - \sum_j \sum_i p_i q_{ij} \lg q_{ij} \\ &= \sum_j \left[\lg \left(\sum_i p_i q_{ij} \right) \right] \sum_i p_i q_{ij} - \sum_i p_i \lg p_i \sum_j q_{ij} - \sum_i p_i \sum_j q_{ij} \lg q_{ij} \\ &= \sum_j \left(\sum_i p_i q_{ij} \right) \lg \left(\sum_i p_i q_{ij} \right) - \sum_i p_i \lg p_i - \sum_i p_i \sum_j q_{ij} \lg q_{ij} \end{aligned}$$

- (2) Ignoring the middle part, which is the pre-question entropy, and is independent of the (parameters of the) question, we are left with

$$\sum_j \left(\sum_i p_i q_{ij} \right) \lg \left(\sum_i p_i q_{ij} \right) - \sum_i p_i \sum_j q_{ij} \lg q_{ij}$$

representing the question’s expected improvement to entropy.

- (3) We can use any base for the logarithm.

- (4) Pre-compute the following for each item i and each question:

$$t_i = \log \sum_j q_{ij} \log q_{ij}$$

- (5) The formula is now equivalent to

$$\sum_j \left(\sum_i p_i q_{ij} \right) \log \left(\sum_i p_i q_{ij} \right) - \sum_i p_i t_i$$

- (6) One can save the $p_i q_{ij}$ to reuse as the updated ranks, after picking a question and getting answer j .
- (7) The first half of this formula can be computed in one pass, by (selecting j and) computing $\sum_i p_i q_{ij}$, for each j .

4. EXAMPLE

Before			Answer 0				Answer 1				t_i
p_i	$\lg p_i$	$p_i \lg p_i$	q_{i0}	p_i^0	$\lg p_i^0$	$p_i^0 \lg p_i^0$	q_{i1}	p_i^1	$\lg p_i^1$	$p_i^1 \lg p_i^1$	
0.30	1.74	0.52	0.9	0.27	1.89	0.51	0.10	0.03	5.06	0.15	0.60
0.20	2.32	0.46	0.1	0.02	5.64	0.11	0.90	0.18	2.47	0.45	0.84
0.10	3.32	0.33	0.5	0.05	4.32	0.22	0.50	0.05	4.32	0.22	1.21
0.10	3.32	0.33	0.5	0.05	4.32	0.22	0.50	0.05	4.32	0.22	1.21
0.10	3.32	0.33	1	0.10	3.32	0.33	0.00	0.00	0.00	0.00	1.59
0.10	3.32	0.33	1	0.10	3.32	0.33	0.00	0.00	0.00	0.00	1.59
0.05	4.32	0.22	1	0.05	4.32	0.22	0.00	0.00	0.00	0.00	2.21
0.05	4.32	0.22	1	0.05	4.32	0.22	0.00	0.00	0.00	0.00	2.21
Σ		2.75	6	0.69	31.46	2.15	2.00	0.31	16.18	1.03	
H		2.75				1.48				0.32	

So the entropy before the question is 2.75 and the expected value after is $1.48 + 0.32 = 1.8$.

5. EXCURSUS

Path Length. Suppose the n leaves of a tree T are assigned probabilities p_i . The *weighted path length* $w(T)$ of T is the weighted average of the length ℓ_i of the paths from the root to the leaves of T :

$$w(T) = \sum_i p_i \ell_i$$

Note. It is known [1] that for all probability distributions P there is a T such that

$$H(P) \leq w(T) \leq H(P) + 2$$

So—for large trees—the entropy is a good estimate of the average path in the best possible tree.¹ (The lower bound follows from the nature of entropy.) The Huffman code algorithm [2] (see below) is optimal.

Theorem 1. *Assume that all probabilities in P are powers 2^{-k} of 2. Then, one can achieve $w(T) = H(P)$.*

Proof. Use Huffman's algorithm:

¹Some other connections between path length and entropy appear in [4, 5].

- (1) Each value in P is a leaf node.
- (2) Replace the two nodes with smallest values, with a node containing their sum, and with the original nodes as its children.
- (3) Repeat previous step until only one node is left.

For the probabilities to add up to 1, the two smallest must be equal; so adding them gives a new power of 2. (Just think of their binary representation.) By induction, every node 2^{-k} will end up on level k . The claim follows. \square

Theorem 2. *One can recursively partition a discrete distribution into two sets of equal total probability, if and only if the probabilities are all powers 2^{-k} .*

Such a recursive partitioning corresponds to a perfectly balanced binary tree, in which the two branches at every node have equal weight.²

Proof. Suppose we are given a set P of n such powers. If $n = 2$, then $P = \{1/2, 1/2\}$, which can be split as desired. Suppose $n > 2$. Replace two equal ones with their sum, which is also of the desired form. By induction, the new, smaller set can be split evenly, after which the two combined probabilities can be decomposed.

Suppose $P = \{p_1, \dots, p_n\}$ can be split as described into P_1, P_2 , each weighing $\frac{1}{2}$. Each of P_j ($j = 1, 2$) can be split recursively, so if we double each probability in P_j , the latter can also be split. Since $|P_1|, |P_2| < |P|$, by induction the doubled probabilities are of the desired form. But then so are the non-doubled ones. \square

Moral. Working under the assumption that there always is some question that splits the relevant set into two halves, the expected length of a path to the chosen (clicked) product for a given query is given by the entropy.

REFERENCES

- [1] Gilbert, E. N., and Moore, E. F. “Variable Length Binary Encodings”, *Bell Sys. Tech. J.* 38 (July 1959), pp. 933–967.
- [2] Huffman, D. A. “A method for the construction of minimum-redundancy codes”, *Proceedings of the I.R.E.*, Sep. 1952, pp. 1098–1102. URL=http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf.
- [3] Quinlan, J. R. “Discovering rules from large collections of examples: a case study”. In D. Michie, editor, *Expert Systems in the Micro-electronic Age*, 168–201. Edinburgh University Press, Edinburgh, 1979.
- [4] Rissanen, J. “Bounds for Weight Balanced Trees”, *IBM J. of R&D* 17 (2), 101–105 (Mar. 1973). URL=<http://www.research.ibm.com/journal/rd/172/ibmrd1702C.pdf>.
- [5] Wong, C. K. and Nievergelt, J. 1973. “Upper Bounds for the Total Path Length of Binary Trees”. *J. ACM* 20, 1 (Jan. 1973), 1-6. DOI=<http://doi.acm.org/10.1145/321738.321739>.

²The Shannon-Fano algorithm works top-down in this way, rather than bottom-up like Huffman.