Coding for General Penalties

Michael Baer
Stanford University
calbear@stanford.edu

Abstract—

Given the probability mass function of an alphabet, Huffman coding finds a corresponding prefix-free binary code that minimizes the expected codeword length. However, there are many practical situations in which the constraints and goals are different from those in Huffman coding; minimization of the linear penalty of expected length may not always be the goal. If the data rate is slow, long codewords are highly undesirable, potentially causing an intolerable delay. Examples for which this is the case include remote exploration, military applications and group testing for diagnostic systems.

In this paper, we examine a family of coding problems with non-linear penalties. We then generalize the most efficient algorithm for finding length-limited codes[12] to an efficient algorithm for finding optimal codes for any penalty in this family, which we denote *general additive convex penalties*. These are penalties of the form $\sum_i f(l_i, p_i)$, where l_i denotes the length of the ith codeword, p_i denotes the corresponding probability and f is convex and increasing in each l_i. This includes several previously proposed penalties, including all those in a general sub-problem proposed by Campbell[3]. In the Campbell case, among others, the optimization may be performed using quadratic time and linear space. This algorithm may also be further applied to an expanded range of penalties.

Keywords— **Optimal prefix code, Huffman algorithm, finite alphabet, coding of integers.**

I. Introduction

A. Problem

The game of twenty questions – in which one is challenged to identify an item by asking up to twenty "yes" or "no" questions – is often cited when explaining the most basic information-theoretic concepts, such as entropy and coding[2]. Indeed, the general problem of binary coding, finding a representation of a possibility in terms of sequences of bits – "yes" or "no" questions – is also that encountered in twenty questions.

However, there is at least one vital difference between standard coding and twenty questions: In the actual game of twenty questions, minimization of mean length is not in fact the goal. The goal instead is to minimize the probability a codeword has more than twenty bits, which can be done by using as many 20-bit codewords as possible. Furthermore, in practical situations the goal may not always be to achieve the optimal average rate. Only in coding for high-rate communications is average rate the precise concern.

Practical problems in which the goal is not minimizing mean length include those involving remote exploration, military applications and group testing for diagnostic systems, e.g. blood testing[10]. In such situations, at least one direction of communication may enable only a handful of crucial bits to be sent – a natural channel may have nearly zero bandwidth, a mission-critical channel may be jammed by an adversary, and blood tests may be costly and time-consuming. Therefore, long codewords should be avoided. In a case in which there are several requests for information, long codewords may also prevent further vital communication through a channel, via the "slow truck effect," the phenomenon by which many small packets (i.e. short codewords or fast cars) are held up by one long packet (long codeword or slow truck)[6]. For these reasons and others, there are cases in which long sequences of bits should be penalized non-linearly.

One of the oldest well-known illustrations of coding is, in fact, a military application, involving a situation said to have occurred during the Bar Kochba revolt of AD 132-135. Rising up against Roman oppression in ancient Palestine, revolution leader Bar Kochba sent out a scout to spy on the Roman camp. The Romans captured and tortured the scout, cutting out his tongue. After escaping, he could not speak and, being illiterate, could not write. He could only answer "yes" or "no" questions by nodding or shaking his head. Bar Kochba used the information gleaned from these questions to defend his fortress.

In Eastern Europe, "twenty questions" was known

as "the Bar-kochba game[15]." However, while the scout's limitations were not to twenty questions, he nevertheless needed to trade off the information gleaned with the time taken to glean it in a manner different from standard coding. To do this, one may use some sort of criterion, or penalty function. The penalty is selected according to the value of obtaining the information in a certain amount of time (or codeword length or number of tests).

In this paper, we concern ourselves with such penalties. We consider a generalization of linear coding which solves a general family of problems.

B. Formalization of the coding problem

Let us briefly formalize the standard coding problem. Each codeword, or sequence of answers to predetermined questions, is in $\{0,1\}^*$, the set of all finite sequences of 0s and 1s (or "yes"s and "no"s).

We wish to assign such binary codewords to a finite alphabet, $\mathcal{X}$. We may assume, without loss of generality, that $\mathcal{X} = \{1, 2, \ldots, n\}$ for some $n = |\mathcal{X}| < +\infty$. The ith member of $\mathcal{X}$ is thus the integer i.

For a given alphabet, random variable $X \in \{1, 2, \ldots, n\}$ is chosen such that $X = i$ with probability $p_i \in (0, 1]$. Probability zero items may be omitted as they never figure in any properties regarding the distribution. Expected values involving X under this probability distribution are referred to as $E_{\mathbf{p}}[f(X)]$, where $\mathbf{p} = (p_1, p_2, \ldots, p_n)$, emphasizing the underlying probability distribution. Without loss of generality, for all $i > j$, $p_i \leq p_j$.

Each item may be assigned a corresponding codeword that we denote as $c_i \in C_{\mathcal{X}} \subset \{0,1\}^*$, where $C_{\mathcal{X}}$, the code, is the collection of codewords chosen. The only fundamental constraint is that the code be prefix-free. That is, to eliminate ambiguity and force codewords to be self-terminating, no codeword may be the prefix of another codeword.

Each codeword c_i is of length l_i, subject to the constraints of the problem. For the sake of notation, we may instead refer to $l(i) = l_i$. A code is feasible if and only if its codeword lengths satisfy the Kraft inequality, $\sum_i 2^{-l_i} \leq 1$, which applies not just to prefix-free codes but to any uniquely decodable code[4]. We will refer to such a set of feasible code lengths as $L_{\mathcal{X}}$.

Definition: A set of lengths, $L_{\mathcal{X}}$, is called a *length ensemble*.

Finding either $C_{\mathcal{X}}$ or $L_{\mathcal{X}}$ solves the problem, as a valid code may easily be found from the length ensemble, and vice versa. Note that when lengths are enumerated in decreasing order and different ensembles for the same n are sorted lexicographically, such sets are well-ordered.

With the Kraft inequality and perhaps other constraints, Huffman coding[8] finds the codebook $C_{\mathcal{X}} = \{c_i\}$ that minimizes the expected length, $E_{\mathbf{p}}[l(X)] = \sum_i p_i l_i$. The central goal of coding theory, due to the strong law of large numbers, is this minimization. With large amounts of data, one need only worry about mean length, and the penalty is linear.

II. A non-linear penalty

A. Formalization and motivation of the problem

We now consider the problem of coding for a penalty that is a non-linear function of codeword lengths. With such a different penalty – for which minimization of mean length is not the goal – the solution to the problem may differ.

The goal should be to minimize a function of the given $\mathbf{p}$ and solution $L_{\mathcal{X}}$ (or $C_{\mathcal{X}}$). A general family of problems to consider are functions of the form $F^{\mathbf{p}}(L_{\mathcal{X}}) = \sum_i f(l_i, p_i)$ for some function $f(x, y) : \mathcal{N} \times [0, 1] \to \mathcal{R} \cup +\infty$. We always assume f to be monotonically increasing in l_i, and usually assume it to be convex in l_i, so that there is a positive penalty for each additional bit, and the penalty for each additional bit is at least equal to that of the last bit. The latter assumption is sensible for examples in which the bits sent are critical.

Given such an f and a $\mathbf{p}$, the problem is then:

$$\begin{array}{ll} \text{Minimize}_{\{L_{\mathcal{X}}\}} & \sum_i f(l_i, p_i) \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathcal{N} \end{array}$$

A sub-problem was proposed by Campbell[3] and covers most interesting cases.[1] He considers Fs of the form $F^{\mathbf{p}}(L_{\mathcal{X}}) = F^{\mathbf{p}}(C_{\mathcal{X}}) := \sum_i p_i f(l_i)$, or $E_{\mathbf{p}}[f(l(X))]$. This problem may be stated as follows:

[1] Campbell also proposes concave and general versions of the problem, but, as previously discussed, here we restrict ourselves to the convex case.

Given probability vector $\mathbf{p}$ and strictly monotonic, convex $f(x)$,

$$\begin{array}{ll} \text{Minimize}_{\{L_{\mathcal{X}}\}} & E_{\mathbf{p}}[f(l(X))] \\ \text{subject to} & \sum_i 2^{-l_i} \leq 1 \\ & l_i \in \mathcal{N}. \end{array} \tag{1}$$

In the linear case, the function is $f(x) = x$; that is, for each additional bit used, we exact a constant penalty.

With the above, we can define a rather general problem:

Definition: Consider $f(l_i, p_i) : \mathcal{N} \times [0,1] \rightarrow \mathcal{R} \cup +\infty$, monotonically increasing and convex with respect to $l_i \in \mathcal{N}$. The *general additive convex coding problem* is that of minimizing $F(L_{\mathcal{X}}, p) := \sum_i f(l_i, p_i)$ over $L_{\mathcal{X}}$, subject to the Kraft inequality and the integer constraint.

This problem is the one we solve here, thus solving the Campbell sub-problem case (1) as well. Heretofore a solution to either case was only known for certain specific cases of this problem. The most general related solution is the distinct one explored in [14], [5], which uses F merely to break ties when the linear case has multiple solutions; the solution uses bottom-merge Huffman coding, first explored in Schwarz[16].

The specific Campbell case of $f(l_i) = \alpha l_i + \beta l_i^2$, for given $\alpha, \beta \geq 0$ has been previously considered by Larmore[11], who presents a $O(n^3)$ time and space algorithm. A version of the algorithm we present can be applied to improve this result to $O(n^2)$ time and linear space.

It is also worthwhile to note that this result of [11] is used in order to construct a polynomial time algorithm for finding a code minimizing a complex non-additive function corresponding to the expected waiting time between information request and receipt.[2] Our results thus improve this algorithm. In fact, they may be used to find efficient polynomial time algorithms to minimize any of a wide variety of functions; see the convex hull theorem of [11] for details.

We assume, without loss of generality, that the domain of f may be extended to $\{\mathcal{N} \cup \{0\}, [0,1]\}$ and that $f(0, p_i) = 0$. If this is not the case, we may replace f with

$$\tilde{f}(l_i, p_i) = \begin{cases} f(l_i, p_i) - 2f(1, p_i) + f(2, p_i), & l_i > 0 \\ f(l_i, p_i) = 0, & l_i = 0 \end{cases}$$

retaining convexity and monotonicity.

We expand upon Larmore and Hirshberg[12] here, making a modification that allows the algorithm to be used for any additive convex penalty. First let us discuss a more general problem, the *Coin Collector's problem.*

B. The Coin Collector's problem and the Package-Merge algorithm

Let $2^{\mathcal{Z}}$ denote the set of all integer powers of two. The Coin Collector's problem of size m considers m coins with width $\rho_i \in 2^{\mathcal{Z}}$; one can think of width as coin face value, e.g. $\rho_i = \frac{1}{4}$ for a quarter. Each coin also has weight $\mu_i \in \mathcal{R}$. The final parameter is total width, denoted T. The problem is thus:

$$\begin{array}{ll} \text{Minimize}_{\{\mathcal{B} \in \mathcal{I}\}} & \sum_{i \in \mathcal{B}} \mu_i \\ \text{subject to} & \sum_{i \in \mathcal{B}} \rho_i = T, \end{array} \tag{2}$$

where $\mathcal{I} := \{1, \ldots, m\}$.

This problem is an input-restricted case of the knapsack problem, which, in general, is $\mathcal{NP}$-hard[13]. However, there is a linear time solution to (2) presented in [12], the Package-Merge algorithm, along with an iterative linear implementation and proof. In our notation, we use $i \in \mathcal{I}$ to denote both the index of a coin and the coin itself.

C. A general algorithm

We now find a reduction from the general additive convex coding problem to the Coin Collector's problem. We first assume bounds on the maximum code length of possible solutions. This may be explicit in the penalty definition as in (3) (below), it may be implicit in some property of the set of optimal solutions, or it may be the maximum unary code length of $n-1$. Then we are only interested in codes with n codewords, none of which has greater length than L for some $\lceil \log_2 n \rceil \leq L \leq n-1$.

Definition: A *node* is an ordered pair of integers (i, l) such that $i \in \{1, \ldots, n\}$ and $l \in \{1, \ldots, L\}$.

[2] The model used assumes information will be sent through a queue with Poisson arrival time and service time proportional to codeword length, i.e. using an M/G/1 queue[6].

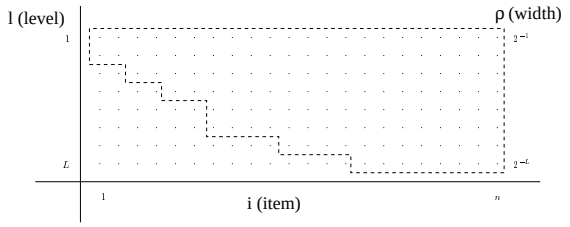

Fig. 1. All possible nodes I and a sample optimal length ensemble nodeset (the subset of I within dashed lines)

Call the set of all $L \cdot n$ possible nodes I (see figure 1). The set of nodes, or *nodeset*, corresponding to a codeword c_i (length l_i) is the set of size l_i, $nodeset(c_i) := \{(j,l) \mid j = i,\ l \in \{1,\ldots,l_i\}\} \subseteq I$. The nodeset corresponding to length ensemble $L_{\mathcal{X}}$ is $nodeset(L_{\mathcal{X}}) := \bigcup_i nodeset(c_i)$, also $\subseteq I$. If node $(i,l) \in I$ then we say it has *width* $\rho(i,l) := 2^{-l}$ and *weight* $\mu(i,l) := f(l,p_i) - f(l-1,p_i)$.

With a simple reduction, any optimal solution S of the Coin Collector's problem for $T = n-1$ on coins $\mathcal{I} = I$ is a nodeset for an optimal solution of the coding problem. Recall $n = |\mathcal{X}|$. Figure 1 shows such a nodeset as a subset of I. Thus this reduction finds an optimal code for all monotonically increasing and convex $f(l_i,p_i)$ in $O(L \cdot n)$ time. The time complexity is dependent on the structure of f and $\mathbf{p}$, ranging from $O(n \log n)$ to $O(n^2 \log n)$ with space requirement $O(n \log n)$ to $O(n^2)$.

D. A general linear space algorithm

Note that the resulting length ensemble need not necessarily have the property that if $i < k$, then $l_i \leq l_k$. For example, if $p_i = p_k$, we are guaranteed no particular inequality relation between l_i and l_k. Also, even if all p_i were distinct, if $f(l_i,p_i) = p_i^{-1}2^{l_i}$, we would expect an inequality relation reversed from the linear case. Thus the problem is quite general, but we can improve upon the algorithm by introducing a reasonable constraint that includes all the cases in the Campbell problem (1).

Definition: A problem is *differentially monotonic* in $\mathbf{p}$ if $\forall\ l > 1,\ p_i > p_k \Rightarrow [f(l,p_i) - f(l-1,p_i)] > [f(l,p_k) - f(l-1,p_k)]$ unless $f(l-1,p_i) = +\infty$.

If the problem is differentially monotonic and all p_i are distinct, we may use a modified version of this algorithm, which we omit for brevity, one using only linear space and $O(n^2)$ time. The modification is a generalization of the linear space algorithm in [12].

However, the assumption of distinct p_i puts an undesirable restriction on our input. Larmore and Hirshberg[12] suggest modifying the probabilities slightly to make them distinct, but this is inelegant and the resulting algorithm is nondeterministic. Here we present an elegant, deterministic alternative, applicable to all differentially monotonic cases.

In initial ordering, we sort the items in reverse of their order of appearance. Recall $\mathbf{p}$ is a decreasing vector. In the first stage of the Package-Merge algorithm, then, combined items are paired off such that all similar-width items in one "package"[3] have adjacent indices. In addition, we choose non-merged items over merged in the case of ties, in the same manner as in the two-queue method of Huffman coding[17]. We obtain a deterministic algorithm retaining this adjacency, and along with it width order preference for items of equal weight, through all steps. Thus we no longer need to worry about a case where $i < k$ but $l_i > l_k$. An additional benefit is that, in the case of ties, this results in minimizing maximal length among optimal codes, as with bottom-merge Huffman coding[16].

E. Examples

The general additive convex coding problem includes such cases as

$$f(l_i,p_i) = p_i l_i^a$$

for $a \geq 1$, the moment penalty, previously with no efficient solution;

$$f(l_i,p_i) = p_i a^{l_i}$$

for $a > 0$, the exponential penalty, previously proposed in Campbell[3] and solved in [7], [9]; and

$$f(l_i,p_i) = \begin{cases} p_i l_i, & l_i \leq l_{max} \\ +\infty, & l_i > l_{max} \end{cases} \tag{3}$$

for some fixed $l_{max} > \lceil \log_2 n \rceil$, the length limited linear penalty, solved efficiently using the Package-Merge algorithm in [12]. This is now a special

[3]Packages of items will be either in the final nodeset or absent from it as a whole.

case of our improved algorithm. All of the above penalties are also Campbell cases (and thus differentially monotonic) and may therefore be solved with quadratic time and linear space.

If we extend the domain of f to a monotonically increasing function on $\{\mathcal{R}^+, [0,1]\}$, we may bound the solution value as follows. Because the optimal solution to the problem without the integer constraint must have at most the same minimum value, this optimal solution, which we denote $l_i^\dagger$ for the ith parameter, provides a lower bound. The Shannon-like code $l_i = \lceil l_i^\dagger \rceil$ provides an upper bound. Thus we have

$$\sum_i f(l_i^\dagger, p_i) \leq \sum_i f(l_i^*, p_i) \leq \sum_i f(\lceil l_i^\dagger \rceil, p_i)$$

where each l_i^* corresponds to the optimal integer length for the ith codeword. Note further that the above upper bound is strictly less than $\sum_i f(l_i^\dagger + 1, p_i)$. These bounds are analogous to Campbell's interpretation of Rényi entropy and may be useful if the real-valued problem can be solved analytically.

F. Conclusion

We have demonstrated an efficient algorithm for general convex coding, with further improvements in space complexity for differentially monotonic cases such as the convex Campbell case.

References

[1] J. Abrahams, "Huffman Code Trees and Variants," *DIMACS Workshop on Codes and Trees: Algorithmic and Information Theoretic Approaches*, Rutgers University, Piscataway, NJ, 1998.

[2] J. Aczél and Z. Darózy, "On Measures of Information and Their Characterizations," Academic, New York, NY, 1975.

[3] L. L. Campbell, "Definition of Entropy by Means of a Coding Problem," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, Vol. 6, pp. 113-118, 1966.

[4] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley-Interscience, New York, NY, 1991.

[5] G. Forst and A. Thorup, "Minimal Huffman trees," *Acta Informatica*, Vol. 36, pp. 721-734, 2000.

[6] R. G. Gallager, *Discrete Stochastic Processes*, Kluwer Academic Publishers, Boston, MA, 1996.

[7] T. C. Hu, D. J. Kleitman, and J. K. Tamaki, "Binary Trees Optimum Under Various Criteria," *SIAM Journal of Applied Mathematics*, Vol. 37, pp. 246-256, 1979.

[8] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. IRE*, Vol. 40, pp. 1098-1101, 1952.

[9] P. A. Humblet, "Generalization of Huffman Coding to Minimize the Probability of Buffer Overflow," *IEEE Transactions on Information Theory*, Vol. IT-27, pp. 230-232, 1981.

[10] S. Kumar and M. Sobel, "Finding a single defective in binomial group testing," *Journal of American Statistical Association*, Vol. 66, pp. 824-828, 1971.

[11] L. L. Larmore, "Minimum Delay Codes," *SIAM Journal on Computing*, Vol. 18, pp. 82-94, 1989.

[12] L. L. Larmore and D. S. Hirshberg, "A Fast Algorithm for Optimal Length-Limited Huffman Codes," *Journal of the Association for Computing Machinery*, Vol. 37, pp. 464-473, 1990.

[13] U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, MA, 1989.

[14] G. Markowsky, "Best Huffman Trees," *Acta Informatica*, Vol. 16, pp. 363-370, 1981.

[15] A. Rényi, *Naplò az információelméletről (A Diary on Information Theory)*, Gondolat, Budapest, Hungary, 1969.

[16] E. S. Schwartz, "An Optimum Encoding with Minimum Longest Code and Total Number of Digits," *Information and Control*, Vol. 7, pp. 37-44, 1964.

[17] J. van Leeuwen, "On the construction of Huffman trees," *Proc. 3rd International Colloquium on Automata, Languages, and Programming*, University of Edinburgh, pp. 382-410, 1976.