Complexity-aware encoding

Yuval Dagan, **Yuval Filmus**, Ariel Gabizon, Shay Moran

23 September 2016

1 Huffman codes and their complexity

Suppose that we are given a distribution μ on the n elements $[n] = \{1, \ldots, n\}$. We want to encode data distributed according to μ using a prefix-free code which minimized the expected codeword size. Denote the cost of the optimal encoding by $T(\pi)$.

Shannon and Fano proved the bounds

$$H(\pi) \leq T(\pi) < H(\pi) + 1,$$

where $H(\pi) = -\sum_i \pi_i \log_2 \pi_i$ is the entropy of π in bits (here $\pi_i = \pi(i)$ is the probability of the ith element). Huffman famously came up with a greedy algorithm that finds an optimal code.

We can describe the encoding process of a prefix-free code by a decision tree, in which each branch is annotated by a bit, and the encoding of an element is formed by concatenating all the annotations from the root to the leaf. The cost of such a decision tree is the expected depth of a leaf, when the leaf is chosen according to π.

Decision trees are ubiquitous in theoretical computer science. Usually a lot of attention is given to the *questions* that can be asked at nodes. From this point of view, Huffman's algorithm is not perfect — the decision tree it constructs could use arbitrary questions. This leads to the following questions:

> Given n, what is the smallest set of questions Q such that for every distribution π on $[n]$ there exists an optimal decision tree for π using only questions from Q?

For reasons that will become clear later on, we denote the smallest number of questions by $u_T(n, 0)$. Obviously, $u_T(n, 0) \leq 2^n$ (in fact, 2^{n-1}). Can we do any better?

Before answering this question, let us rephrase it. The first observation is that it suffices to consider *dyadic* distributions π, which are ones in which $\pi_i = 2^{-k_i}$. For such distributions, we demand there to exist a decision tree in which node i has depth k_i. Why is this reduction valid? Roughly speaking, when constructing a Huffman tree, we are "rounding" the given distribution to a dyadic one.

Second, a set of questions works on all dyadic distributions if and only if for every dyadic distribution π there is a question Q such that $\pi(Q) = 1/2$; we say that Q *splits* π. This is because an optimal tree for π must split π and its marginals at every single node.

We are now ready to present a simple lower bound on $u_T(n, 0)$. Let α be a parameter to be chosen later on. We will consider dyadic distributions for which the optimal decision tree looks like this: take a complete binary tree consists of αn nodes (so this works only if αn is a power of 2), and replace one node with a path of length $(1 - \alpha)n$. We can choose the names of the leaves arbitrarily.

The only questions splitting such a distribution are ones which contain *exactly* $(\alpha/2)n$ of the shallow leaves, as well as the complements of these questions. Thus in order to handle all such distributions, the number of questions we need is

$$\frac{\#\text{ distributions}}{\#\text{ distributions/question}} = \frac{\#\text{ questions}}{\#\text{ questions/distribution}} = \frac{\binom{n}{(\alpha/2)n}}{\binom{\alpha n}{(\alpha/2)n}} \approx \frac{2^{h(\alpha/2)n}}{2^{\alpha n}} = 2^{(h(\alpha/2)-\alpha)n}.$$

Here the first equation follows double counting the pairs consisting of a distribution and a question splitting it. Taking $\alpha = 1/5$, this gives a lower bound of 1.25^n.

Amazingly, this lower bound is attainable! The construction proceeds roughly as follows. For each dyadic distribution π, we find a large set of questions that split it. We then construct a hitting set for *all* dyadic distributions, using a simple randomized construction.

Let a distribution π be given. We arrange the elements in decreasing order of probability, and put them in bins according to their probability: A_k contains those elements whose probability is 2^{-k}. We pool together all small elements into a new synthetic element of total probability 2^{-m}, where A_m is the first non-empty bin. A similar pooling together ensures that all the bins have an even number of elements (though some of these elements might be artificial). Suppose that altogether, we have αn remaining elements.

Each of the following questions splits π: choose half of the elements out of each non-empty bin A_k. Simple estimates show that there are only $O(\log n)$ bins, and so this results in roughly $2^{\alpha n}$ questions, a polynomial fraction of them consisting of exactly $(\alpha/2)n$ elements.

We construct a hitting set valid for all distributions π by choosing a roughly $2^{-\alpha n}$ fraction of questions of size $(\alpha/2)n$ for each α. This results in roughly this number of questions:

$$\max_{\alpha \in [0,1]} \frac{2^{h(\alpha/2)n}}{2^{\alpha n}} = 1.25^n.$$

This non-constructive bound shows that 1.25^n questions suffice to match the performance of Huffman codes. It leaves two important questions open:

1. Is there an explicit set of 1.25^n questions that works?
2. The bound 1.25^n is only tight when $n/5 \approx 2^m$. What happens for other n? We conjecture that the optimal number of questions depends only on the fractional part of $\log_2 n$.

While we have demonstrated significant savings, recovering the performance of Huffman codes exactly requires an exponential number of questions. What if we allow a simple error? Let $u_T(n, r)$ the minimum number of questions needed to match the performance of Huffman codes up to an additive error term of r (we concentrate on $r \leq 1$, for now).

Using two different algorithmic approaches, we can show that $u_T(n, r) = \Theta(n^{\Theta(1/r)})$, and that $u_T(n, 1/2) = O(n^2)$; thus allowing an arbitrarily small constant error reduces the number of questions from exponential in n to polynomial in n.

2 Comparing to the entropy

As we have seen above, the cost of an optimal code for a distribution π is bounded by $H(\pi) + 1$. This bound is achieved by distributions concentrated on a single element x. Such distributions have entropy roughly zero, but identifying x still requires asking at least one questions.

This suggests looking for a small set of questions that suffices to obtain the bound $H(\pi) + 1$. For reasons that the reader can now guess, we denote the smallest number of such questions by $u_H(n, 1)$.

A natural set of questions that arises in practice are *comparisons*: the $n - 1$ questions of the form "$x \leq k$" for $1 \leq k \leq n - 1$. Decision trees using this set of questions are also known as *binary search trees*. Binary search trees cannot achieve $H(\pi) + 1$: the distribution $\epsilon, 1 - 2\epsilon, \epsilon$ has very small entropy, but two comparisons are required to isolate the second element.

Gilbert and Moore showed that the bound $H(\pi)+2$ is always achievable for binary search trees, using an algorithm reminiscent of arithmetic coding. An even simpler algorithm was suggested by Rissanen, and was shown by Horibe to also attain the bound $H(\pi)+2$. This algorithm, known as *weight balancing*, might be the first one that comes to mind:

> Ask the question that splits the domain as evenly as possible, and recurse.

The hard distributions outlined above can be handled by *equality* queries, questions of the form "$x = k$"; in fact, any set of questions achieving $H(\pi) + 1$ must contain all of these questions. A decision tree in which nodes are either comparisons or equality tests are known as *binary comparison search trees*,

but their performance hasn't been evaluated in the literature (people have concentrated on algorithms that find the best trees). We show that such trees achieve the bound $H(\pi)+1$, using the following simple modification of weight balancing:

> If the maximum probability element has probability at least 0.3, ask about it.
> Otherwise, ask the question that splits the domain as evenly as possible, and recurse.

The threshold 0.3 can be fiddled with a bit, but is not completely arbitrary. Unfortunately, the proof is too long to describe here.

Why is the distinction between $H(\pi)+1$ and $H(\pi)+2$ so important? Suppose that we want to reduce the number of questions even further, and we are willing to increase the resulting error. In order to achieve an additive error of r, we need our collection of questions to single out any element using at most r questions (this comes from considering distributions concentrated on that element), and so $\binom{2Q}{\leq r} \geq n$, which implies that we need $\Omega(rn^{1/r})$ questions.

Our modified weight balancing algorithm shows that this bound is achievable! We can think of the number $1,\ldots,n$ as consisting of r digits in base $n^{1/r}$. Using the chain rule for entropy, we see that using questions of the form $x_i \leq k$ and $x_i = k$, we can determine the digits one by one at an additive cost of 1 per digit, and r overall. This uses $O(rn^{1/r})$ questions.

While it seems that our results are quite tight, there is still room for improvement:

1. How many questions are needed to obtain $H(\pi)+1$? We need at least n, and modified weight balancing uses $2n-3$.

2. How many questions are needed to obtain $H(\pi)+r$? A strengthened form of the lower bound outlined above shows that $(1/e)rn^{1/r}$ questions are necessary, and the algorithm we outlined uses $2rn^{1/r}$ questions.

3 Witness codes

When can a set of questions Q be used to single out each element using at most r questions? If we identify each element with a vector of length Q, we need there to be r columns which identify it from all other vectors. We call these columns the *witness* of the vector.

An (n,w) *witness code* is a set of vectors in $\{0,1\}^n$, each of which has a witness of size at most w. How big can an (n,w) witness code be? Taking all vectors of weight w gives a witness code with $\binom{n}{w}$ vectors, and the polynomial method gives an upper bound of $\binom{n}{\leq w}$. Which bound is right? The conjecture is that for each w, if n is large enough then $\binom{n}{w}$ is the correct bound.