CS 395T: Sublinear Algorithms Fall 2014

Lecture 16 — Oct 21, 2014

Prof. Eric Price *Scribe: Chi-Kit Lam*

1 Overview

In this lecture we will talk about information and compression, which the Huffman coding can achieve with the average number of bits sent almost precisely equal to the entropy.

Then we will talk about communication complexity and information cost, which is the asymptotic number of bits two parties need to transmit in a conversation in order to compute some function of their inputs.

We will also talk about adaptive sparse recovery, which is like the conversation version of sparse recovery.

2 Information and compression

2.1 Entropy

Suppose we have a distribution P over the letters. We want to know how many bits we need to send. In particular, we want a prefix code $x \to c_x \in \{0,1\}^*$, which is basically a binary tree. In the example in Figure **??**, the string `TAN` is encoded as 10010111011. We want to minimize the average length $\mathbb{E}_{x\sim P} |c_x|$, which we will call ℓ.

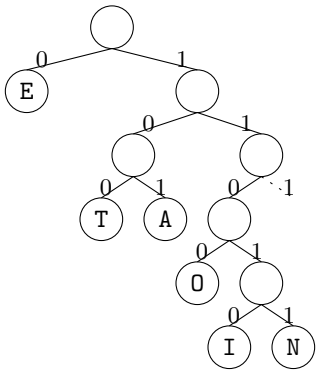

Figure 1: A binary tree corresponding to a prefix code.

Theorem 1 ([**?**])**.** *$\ell \geq H(P)$, where $H(P) = \sum_x P(x) \log_2 \frac{1}{P(x)} = \mathbb{E}_{x\sim P} \log_2 \frac{1}{P(x)}$ is the entropy.*

Huffman[**?**] gave a simple optimal construction of a prefix code with $\ell < H(P) + 1$.

Suppose

$$p = \begin{cases} 1 & \text{with probability } 1/100 \\ 0 & \text{with probability } 99/100 \end{cases}.$$

Then $H(p) = \frac{1}{100}\log 100 + \frac{99}{100}\log\frac{100}{99} \approx \frac{7}{100} + \frac{1}{100} \approx \frac{8}{100}$. In this case, Huffman's code still needs about 1 bit. So, the entropy is essentially "rounded up".

In general, if $x_1, \ldots, x_n \sim P$, then we can encode $(x_1, \ldots, x_n)$ in at most $H(x_1, \ldots, x_n) + 1 = nH(P) + 1$ bits, while it cannot be done in less than $nH(P)$ bits. This is a prettily strong "direct sum" statement. Solving 1 copy costs $H(P) \le \ell < H(P) + 1$, while solving n copies costs on average $H(P) \le \ell < H(P) + 1/n$. This shows that one can gain just a little but not much by doing multiple copies.

2.2 Huffman coding

The algorithm of Huffman coding is actually quite simple. Given the distribution, the algorithm sorts the letters by frequency. Then, the last two letters are merged and their frequencies are summed up into a new node. This process repeats until everything is merged together, resulting a binary tree. Figure **??** shows the first two iterations of the algorithms.

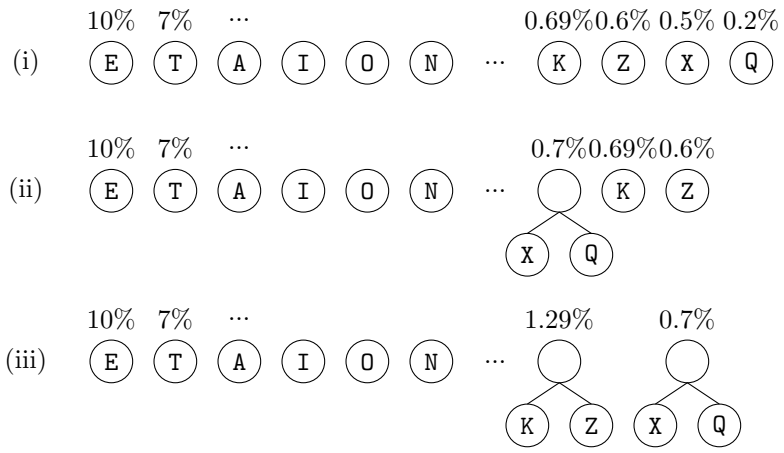

Figure 2: The first two iterations of the algorithm. (i) Initially, the letters are sorted by frequency. (ii) In the first iteration, `X` and `Q` are merged. (iii) In the second iteration, `K` and `Z` are merged.

We can show that $|c_x| < \log\frac{1}{P(x)} + 1$. (Intuitively, if $P(x) = 2^{-k}$, then there are at most 2^k nodes with larger frequencies and they can fit into a complete binary tree. So, $|c_x| \le k$.) This implies $\ell = \mathbb{E}\,|c_x| < H(P) + 1$.

3 Communication Complexity and Information Cost

3.1 Communication Complexity

In the communication setting, Alice has some input x and Bob has some input y. They share some public randomness and want to compute $f(x, y)$. Alice sends some message m_1, and then Bob

responds with m_2, and then Alice responds with m_3, and so on. At the end, Bob outputs $f(x, y)$. They can choose a protocol Π, which decides how to assign what you send next based on the messages you have seen so far and your input. The total number of bits transfered is $|\Pi| = \sum |m_i|$.

The communication complexity of the protocol Π is

$$CC_\mu(\Pi) = \mathop{\mathbb{E}}_{\mu}(|\Pi|),$$

where μ is a distribution over the inputs (x, y) and the protocol. The communication complexity of the function f for a distribution μ is

$$CC_\mu(f) = \min_{\Pi \text{ solves } f \text{ with 3/4 prob}} CC_\mu(\Pi).$$

The communication complexity of the function f is

$$CC(f) = \max_{\mu} CC_\mu(f).$$

3.2 Information Cost

Information cost is related to communication complexity, as entropy is related to compression.

Recall that the entropy is $H(X) = \sum p(x) \log \frac{1}{p(x)}$. Now, the mutual information $I(X;Y) = H(X) - H(X|Y)$ between X and Y is how much a variable Y tells you about X. It is actually interesting that we also have $I(X;Y) = H(Y) - H(Y|X)$.

The information cost of a protocol Π is

$$IC(\Pi) = I(X; \Pi|Y) + I(Y; \Pi|X).$$

This is how much Bob learns from the protocol about X plus how much Alice learns from the protocol about Y. The information cost of a function f is

$$IC(f) = \min_{\Pi \text{ solves } f} IC(\Pi).$$

For all protocol Π, we have $IC(\Pi) \leq \mathbb{E}\,|\Pi| = CC(\Pi)$, because there are at most b bits of information if there are only b bits transmited in the protocol. Taking the minimum over all protocols implies $IC(f) \leq CC(f)$. This is analogous to Shannon's result that $H \leq \ell$.

It is really interesting that the asymptotic statement is true. Suppose we want to solve n copies of the communication problem. Alice given $x_1, \ldots, x_n$ and Bob given $y_1, \ldots, y_n$, they want to solve $f(x_1, y_1), \ldots, f(x_n, y_n)$, each failing at most $1/4$ of the time. We call this problem the direct sum $f^{\oplus n}$. Then, for all functions f, it is not hard to show that $IC(f^{\oplus n}) = nIC(f)$.

Theorem 2 ([?])**.**

$$\frac{CC(f^{\oplus n})}{n} \to IC(f) \qquad \textit{as } n \to \infty.$$

In the limit, this theorem suggests that information cost is the right notion.

3.3 Separation of Information and Communication

The remaining question is, for a single function, whether $CC(f) \approx IC(f)$, in particular whether $CC(f) = IC(f)O(1) + O(1)$. If this is true, it would prove the direct sum conjecture $CC(f^{\oplus n}) \gtrsim nCC(f) - O(1)$.

The recent paper by Ganor, Kol and Raz [**?**] showed that it is not true. They gave a function f for which $IC(f) = k$ and $CC(f) \geq 2^{\Omega(k)}$. This is the best because it was known before this that $CC(f) \leq 2^{O(IC(f))}$. The function that they gave has input size $2^{2^{2^k}}$. So, it is still open whether $CC(f) \lesssim IC(f) \log\log|\text{input}|$.

A binary tree with depth 2^{2^k} is split into levels of width $\approx k$. For every node v in the tree, there are two associated values x_v and y_v. There is a random special level of width $\approx k$. Outside this special level, we have $x_v = y_v$ for all v. We think about x_v and y_v as which direction you ought to go. So, if they are both 0, you want to go in one direction. If they are both 1, you want to go in the other. Within the special level, the values x_v and y_v are uniform. At the bottom of the special level, v is *good* if the path to v is mostly (80%) following directions. The goal is to agree on any leaf v' where v' is a descendent of some good vertex.

What makes it tricky is that you do not know where the special level is, because if you knew where the special level was, then $O(k)$ communication suffices. The problem is you do not know where the special level is. You can try binary searching to find the special level, taking $O(2^k)$ communication. This is basically the best you can do apparently.

We can construct a protocol with information cost only $O(k)$. It is okay to transmit something very large as long as the amount of information contained in it is small. Alice can transmit her path and Bob just follows it, and that is a large amount of communication but it is not so much information because Bob knows what the first set would be. The issue is that it still gives you $\approx 2^k$ bits of information knowing where the special level is. The idea is instead that Alice chooses a noisy path where 90% of the time follows her directions and 10% deviates. This path is transmitted to Bob. It can be shown that this protocol only has $O(k)$ information.

Therefore, many copies can get more efficient.

4 Adaptive Sparse Recovery

Adaptive sparse recovery is like the conversation version of sparse recovery.

In non-adpative sparse recovery, Alice has $i \in [n]$ and sets $x = e_i + w$. She transmits $y = Ax = Ae_i + w'$. Bob receives y and recovers $y \to \hat{x} \approx x \to \hat{i} \approx i$. In this one-way conversation, we showed

in last class that

$$\begin{aligned} I(\hat{i};i) \leq & I(y;i) \\ \leq & m(0.5\log(1+\mathrm{SNR})) \\ \lesssim & m \\ H(i) - H(i|\hat{i}) \lesssim & m \\ \log n - (0.25\log n + 1) \lesssim & m \\ m \gtrsim & \log n. \end{aligned}$$

In the adaptive case, we have something more of a conversation. Alice knows x. Bob sends v_1 and Alice sends back $\langle v_1, x\rangle$. Then, Bob sends v_2 and Alice sends back $\langle v_2, x\rangle$. And then, Bob sends v_3 and Alice sends back $\langle v_3, x\rangle$, and so on.

To show a lower bound, consider stage r. Define P as the distribution of $(i|y_1, \ldots, y_{r-1})$. Then, the observed information by round r is $b = \log n - H(P) = \mathbb{E}_{i\sim P} \log(np_i)$. For a fixed v depending on P, as $i \sim P$, we know that

$$I(\langle v, x\rangle ; i) \leq \frac{1}{2}\log\left(1 + \frac{\mathbb{E}_{i\sim P}\, v_i^2}{\|v_i\|_2^2/n}\right).$$

With some algebra (Lemma 3.1 in [**?**]), we can bound the above expression by $O(b+1)$. It means that on average the number of bits that you get at the next stage is $\lesssim 2$ times what you had at the previous stage. This implies that R rounds take $\Omega(R\log^{1/R} n)$ measurements. And in general, it takes $\Omega(\log\log n)$ measurements.

References

[BR11] M. Braverman, A. Rao. Information Equals Amortized Communication. *FOCS* 2011: 748-757

[GKR14] A. Ganor, G. Kol, R. Raz. Exponential Separation of Information and Communication. *ECCC*, Revision 1 of Report No. 49 (2014)

[Huf52] D. A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE, vol.40, no.9, pp.1098,1101, Sept. 1952

[PW13] Eric Price, David P. Woodruff. Lower Bounds for Adaptive Sparse Recovery. *SODA* 2013: 652-663.

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal* vol.27, no.4, pp.623-656, Oct. 1948