

Braverman-Rao Protocol and Some Implications

Scribed by *Xi Wu*

Contents

1	Introduction	2
2	Main Results	2
3	Sampling Elements on Public Randomness	2
4	Intuitions	3
5	Issues of the Protocol and How to Fix	4

1 Introduction

Consider the following information transmission problem between Alice and Bob. Alice is given a distribution P and Bob is given a distribution Q over a common universe. They do *not* know the distribution on the other side but they have access to a shared randomness. Now Alice draws an element a from P and wants to let Bob know it. The problem is to minimize the transmission cost (in terms of number of bits).

In the upcoming FOCS, Mark Braverman and Anup Rao give an *interactive*, public-coin protocol that for each element $a \sim P$, with probability at least $1 - \epsilon$, Bob will learn a , and the communication cost is $\log P(a)/Q(a)$ plus some *sublinear term* which depends on $1/\epsilon$. An immediate observation is that this protocol would break the barrier of entropy: when P and Q are near to each other, the cost is nearly zero. However, this protocol heavily relies on interaction (and also public randomness). It remains an open problem whether interaction is truly necessary.

There are several interesting implications with this protocol. The first immediate implication is that we have a protocol with expected cost $\sum_a P(a) \log \frac{P(a)}{Q(a)}$ plus some sublinear term, which is roughly $\mathbf{D}(P||Q)$, the *information divergence* between P and Q . Second, one can *compress* any k -round protocol by using the low-cost protocol for one element k times with error $k\epsilon$, and the communication complexity of the compressed protocol is roughly bounded by an information theory term defined on the protocol tree (so called *divergence cost* of the protocol tree). Finally, let $f : X \times Y \mapsto Z$ be a function computed by protocol π and U be a distribution on $X \times Y$, by applying this compression method to simulate n copies of π , we find that, when n tends to infinity, the communication complexity averaged by n tends to internal information cost of computing f according to U (another information theory term we will formally define). The last implication is a bit surprising as it relates two seemingly unrelated quantities.

2 Main Results

Theorem 1. *Suppose that player A is given a distribution P and player B is given a distribution Q over a universe U . There is a protocol such that at the end of the protocol:*

- *player A outputs an element a according to distribution P ;*
- *player B outputs an element b such that for each x , $\Pr[b = a \mid a = x] > 1 - \epsilon$.*
- *the communication in the protocol is bounded by*

$$\log P(a)/Q(a) + \log 1/\epsilon + \log \log 1/\epsilon + 5\sqrt{\log P(a)/Q(a)} + 9$$

3 Sampling Elements on Public Randomness

Consider the distribution P on point set \mathcal{U} . The paper uses the following procedure to sample an element according to P . Let $p(x_i)$ be the probability of x_i according to distribution P .

1. Uniformly and independently sample elements from $a_i = (x_i, p_i) \in \mathcal{U} \times [0, 1]$, for $i = 1, 2, \dots$
2. Pick the first element a_i that $p_i < p(x_i)$, let this element be the output a .

We use $a[0]$ to denote its first coordinate and $a[1]$ to denote its second coordinate. The claim is that:

Proposition 1. $\Pr[a[0] = x_i] = p(x_i)$

That is the distribution of a is exactly the distribution of x_i . Intuitively this is true, because if $p(x_i)$ is big, then it has a larger chance that $p_i < p(x_i)$.

Proof. $\Pr[a[0] = x_i] = \Pr[a[1] < p(x_i) \mid a[0] = x_i] = \frac{p(x_i)}{1} = p(x_i)$ □

Note in the proof we need the sampling is independent – therefore whether we pick a_i or not completely depends of $p(x_i)$ (we uniformly pick $p_i \in [0, 1]$). This gives a very concrete way to sample elements according to P – and all we have to do is just uniformly sampling.

4 Intuitions

The intuition of the protocol is indeed (in my eyes), quite related to the sampling method described above, as we will see in the rest of this section.

At the very start, a sequence of elements are uniformly sampled from $\mathcal{U} \times [0, 1]$ and are put on the public randomness (this step seems a bit cheating to me, though). For now, think the sequence consists of infinitely many elements. Now Alice picks the first element (x_i, p_i) that $p_i \leq P(x_i)$. We will only work on the range of $|\mathcal{U}|$ elements containing x_i , this can be done by sending only constant number of bits: see Section 5 for more details. From now on, denote this element by (α, p) . Note that $p < P(\alpha)$.

Because that all the elements are actually on the public randomness, hence it is natural to use hashing to reduce the number of bits transmitted. Suppose Alice and Bob share a family of universal binary hashing functions $H = \{h \mid h : \mathcal{U} \mapsto \{0, 1\}\}$ (again put on the public randomness). Hence for every *fixed* x , we have that:

$$\Pr_{h \sim H} [h(x) = h(\alpha)] = 1/2$$

The most straightforward idea would be, Alice sends Bob hash bits incrementally one bit by one bit. And if in one round, Bob finds for one element x , all the hash values matches, then he outputs x . Let's look at the case for one fixed x . After t rounds, the probability that Bob mistakenly outputs $x \neq \alpha$ for any fixed x would be $1/2^t$. However, here we will have a union bound on t , and summing up $\sum_{i \geq 1} 1/2^i$ is 1, no bound at all.

This indicates that we should have a bootstrapping number of hash values. Suppose we want to reach error probability ϵ for x . Then in the first round, Alice sends $\log \frac{2}{\epsilon}$ hash bits. Then the error probability in the first round is $(1/2)^{\log \frac{2}{\epsilon}}$. Then Alice sends incrementally one bit by one bit, this will give a bound:

$$\sum_{i \geq 1} \frac{\epsilon}{2^i} = \epsilon \tag{1}$$

Sounds good. However this is still not enough. Because \mathcal{U} may have many elements, hence the total error probability, when applying union bound, will give us an error $|\mathcal{U}|\epsilon$, this indicates that if we want to bound this by a constant ϵ' , we must set ϵ to $\frac{\epsilon'}{|\mathcal{U}|}$, which means in the first round Alice

needs to send $\log \frac{2|\mathcal{U}|}{\epsilon} > \log |\mathcal{U}|$ bits: even worse than the naive protocol of sending α itself (costs $\log |\mathcal{U}|$ bits).

Clearly we do not want to send so many hash bits. So let's step back and stick to the bootstrapping number being $\log \frac{2}{\epsilon}$. However, now we want to do a twist so that the error probability would become $\frac{\epsilon}{2|\mathcal{U}|}$, seems impossible. Actually this is again where the protocol is closely related to the sampling method.

Observe that the elements sampled on public randomness are uniformly distributed on a square of area $|\mathcal{U}|$. And Bob has a distribution Q of which the area is only 1. Therefore if Bob restricts his eyes to those elements that fall into his distribution, the probability that an element falls into distribution Q is only $1/|\mathcal{U}|$. More precisely, Bob now looks at those elements where $p_j \leq Q(x_j)$, and compare hash bits with those Alice sends *only on these elements*. This means we will take into account the probability space public randomness uses. For any fixed element on the public randomness, the probability that Bob mistakenly outputs it at the bootstrapping step is now:

$$\left(\frac{1}{2}\right)^{\log \frac{2}{\epsilon}} \cdot \frac{1}{|\mathcal{U}|} = \frac{\epsilon}{2|\mathcal{U}|} \quad (2)$$

Good, now we can apply union bound. But this immediately brings us an issue: (α, p) may not be in distribution Q at all. Specifically if $P(\alpha) \geq p > Q(\alpha)$, then Bob will never try to look at α !

Now it remains to handle the case that $P(\alpha) > Q(\alpha)$. It is now natural to gradually expand distribution Q , until α falls into the eyes of Bob. Specifically, at round t , we will use the distribution $2^t \cdot Q$. This is natural, because roughly we want to let Bob learn $\log P(\alpha)/Q(\alpha)$, and at round $\log P(\alpha)/Q(\alpha)$, Bob will definitely start to look at α as $2^{\log P(\alpha)/Q(\alpha)} Q(\alpha) = P(\alpha)$. Now the above argument changes a little bit, the probability that we will look at an element becomes $2^t/|\mathcal{U}|$, and hence for canceling purpose, we can set

$$\left(\frac{1}{2}\right)^{2t + \log \frac{2}{\epsilon}} \cdot \frac{2^t}{|\mathcal{U}|} = \frac{\epsilon}{2^t |\mathcal{U}|}$$

That is in every round we send 2 bits (rather than one at a round), besides the bootstrapping number of hash values. This counts *in total* $2t + \log \frac{2}{\epsilon}$ hash values. By union bound this gives us error probability at most ϵ . Simple counting shows us that the total communication cost is $c \cdot \log P(\alpha)/Q(\alpha)$, for some constant $c > 1$.

A little more twist allows us to get the bound stated in the theorem. Consider we grow the distribution even faster. Say, at double exponential rate 2^{t^2} . This makes more elements fall into the eyes of Bob – but then to reduce the error back to $O(\epsilon/|\mathcal{U}|)$, we only to send *more hash values*. Simple calculation shows that setting to $(t+1)^2$ suffices: this will give us a protocol of cost $\log P(\alpha)/Q(\alpha)$, suppressing constant and sublinear terms.

5 Issues of the Protocol and How to Fix

The protocol we presented in previous sections almost works except some subtle issues. The main one is that how could we write down, in the public randomness, infinitely many elements uniformly and independently sampled from $\mathcal{U} \times [0, 1]$.

The way we fix it follows a quite standard and natural idea: Recall that Alice outputs the first element that falls in to distribution P . We show that with high probability, this element will appear with a small number of samples. Then we can apply union bound by ignoring the case

where we need huge number of samples to get an element from P . Remember that we send the index $k = i/|\mathcal{U}|$, formally we show the following:

Lemma 1. $\Pr[k > n] < e^{-n}$

Proof. Using the area method, the whole sample space has area $1 \cdot |\mathcal{U}|$ and the distribution P has area 1. Therefore with uniform sampling in $\mathcal{U} \times [0, 1]$, the probability that we do not sample an element in P is:

$$1 - \frac{1}{|\mathcal{U}|}$$

With $k > n$, therefore we have sampled more than $k \cdot |\mathcal{U}| = n|\mathcal{U}|$ elements, which means the probability is bounded by (as all samples are independent)

$$\left(1 - \frac{1}{|\mathcal{U}|}\right)^{n|\mathcal{U}|} < e^{-n}$$

□

Note in the protocol, we use roughly $\log \log 1/\epsilon$ bits to encode k , this means the probability that we cannot use these many bits to encode k would be small:

$$e^{-2^{\log \log 1/\epsilon}} = e^{-\log 1/\epsilon} = e^{\log \epsilon} = \epsilon$$