Chapter 10

Conditional entropy re-visited

RECALL THAT in Chapter 4 we defined the conditional entropy $H(s \mid t)$ of one string s given another string t. We stated, but did not prove, the fundamental identity

$$H(s,t) = H(t) + H(s \mid t).$$

Informally, this says that the information in t, together with the additional information in s, is precisely the information in both s and t. This result is the last piece of the jigsaw in the basic theory of algorithmic entropy. The proof we give now is somewhat convoluted, involving the equivalence of the two entropies, as well as a further application of the converse to Kraft's Inequality.

10.1 Conditional Entropy

Recall the definition of $H(s \mid t)$: Let $\tau = \mu(t)$ be the minimal input outputting t from our universal machine U. Then we consider the set of pairs (T, p) consisting of a Turing Machine T and a string p such that

$$T(\tau p) = s;$$

and we define $H(s \mid t)$ to be the minimum of

$$|\langle T \rangle| + |p|.$$

(We cannot simply take the minimum of $|p|$ over strings p such that

$$p : U(\tau p) = s,$$

because U is going to halt as soon as it has read in τ. So we make an indirect reference to the fact that

$$H(s) \leq H_T(s) + |\langle T \rangle|,$$

since $U(\langle T \rangle p) = T(p)$.)

10.2 The last piece of the jigsaw

Theorem 10.1.

$$H(s,t) = H(t) + H(s \mid t) + O(1).$$

Our proof is in two parts.

1. The easy part:

$$H(s,t) \leq H(t) + H(s \mid t) + O(1).$$

2. The harder part:

$$H(s \mid t) \leq H(s,t) - H(t) + C,$$

where C is a constant depending only on our choice of universal machine U.

10.3 The easy part

Proposition 10.1. *There exists a Turing machine M such that*

$$H_M(s,t) \leq H(t) + H(s \mid t)$$

for all $s, t \in \mathbb{S}$.

Proof ▶. Let

$$\tau = \mu(t)$$

be the minimal program for t:

$$U(\tau) = t, \quad H(t) = |\tau|.$$

By the definition of conditional entropy $H(s \mid t)$, we can find a machine T and a string p such that

$$T(\tau p) = s, \quad H(s \mid t) = |\langle T \rangle| + |p|.$$

It follows that

$$U(\langle T\rangle \tau p) = T(\tau p) = t.$$

Now we construct the machine M as follows. M starts by imitating U, so that if the input string is

$$q = \langle T\rangle \tau p$$

then M will compute $U(q) = t$. However, it does not output t but simply records its value for later use.

But now M goes back to the beginning of the string q (which it has wisely stored) and skips the machine code $\langle T\rangle$.

Now M imitates U again, but this time with input τp. Since $U(\tau) = s$, U will only read in the prefix τ before halting and outputting s. Our machine M does not of course output s. Instead it outputs the coded version $\langle s\rangle$.

Finally, it goes back to the remembered string t and outputs $\langle t\rangle$ before halting.

In summary,

$$M(\langle T\rangle \tau p) = \langle s\rangle\langle t\rangle.$$

It follows that

$$\begin{aligned} H_M(s,t) &\le |\langle T\rangle| + |\tau| + |p| \\ &= |\tau| + (|\langle T\rangle| + |p|) \\ &= H(t) + H(s \mid t). \end{aligned}$$

Corollary 10.1. *We have*

$$H(s,t) \le H(t) + H(s \mid t) + O(1).$$

Proof ▶. Since

$$H(s,t) \le H_M(s,t) + |\langle M\rangle|,$$

the Proposition implies that

$$\begin{aligned} H(s,t) &\le H(t) + H(s \mid t) + |\langle M\rangle| \\ &= H(t) + H(s \mid t) + O(1). \end{aligned}$$

10.4 The hard part

Proposition 10.2. *For each string $t \in \mathbb{S}$,*

$$\sum_{s\in\mathbb{S}} 2^{-(h(s,t)-h(t))} \le C,$$

where C is a constant depending only on our choice of universal machine U.

Proof ▶. **Lemma 1.** *Given a machine T, there exists a machine M such that*

$$\sum_{s\in\mathbb{S}} \mathrm{P}_T(s,t) \leq \mathrm{P}_M(t)$$

(where $\mathrm{P}_T(s,t) = \mathrm{P}_T(\langle s\rangle\langle t\rangle)$).

Proof ▶. Let the machine M start by imitating T, except that instead of outputting $\langle s\rangle\langle t\rangle$, it skips $\langle s\rangle$ and then decodes $\langle t\rangle$ —that is, as T outputs $\langle s\rangle\langle t\rangle$ M outputs t, and then halts.

It follows that

$$T(p) = \langle s\rangle\langle t\rangle \Longrightarrow M(p) = t.$$

Hence

$$\begin{aligned}\bigcup_{s\in\mathbb{S}} \mathrm{P}_T(s,t) &= \sum_{s\in\mathbb{S}} \left(\sum_{p:T(p)=\langle s\rangle\langle t\rangle} 2^{-|p|} \right) \\ &\leq \sum_{p:M(p)=t} 2^{-|p|} \\ &= P_M(t).\end{aligned}$$

Lemma 2. *With the same assumptions as the last Lemma,*

$$\sum_{s\in\mathbb{S}} 2^{-h_T(s,t)} \leq 2^{-h_M(t)}.$$

Proof ▶. This follows at once from the last Lemma, since

$$2^{-h_T(s,t)} = \mathrm{P}_T(s,t), \quad 2^{-h_M(t)} = \mathrm{P}_M(t).$$

Lemma 3. *For any Turing machine T,*

$$h_T(s) \leq 2^{|\langle T\rangle|} h(s).$$

Proof ▶. Since

$$T(p) = s \Longleftrightarrow U(\langle T\rangle p) = s,$$

it follows that

$$\begin{aligned}2^{-h(s)} &= \sum_{q:U(q)=s} 2^{-|q|} \\ &\geq \sum_{p:T(p)=s} 2^{-|\langle T\rangle p|} \\ &= 2^{-|\langle T\rangle|} \sum_{p:T(p)=s} 2^{-|p|} \\ &= 2^{-|\langle T\rangle|}\ h(s).\end{aligned}$$

Lemma 4.

$$\sum_{s\in\mathbb{S}} 2^{-h(s,t)} \leq 2^{-h(t)+c},$$

where c is a constant depending only on our choice of universal machine U.

Proof ▶. This follows at once on taking $T = U$ in Lemma 2, and applying Lemma 3 with $T = M$.

The result follows on taking $h(t)$ to the other side.

Corollary 10.1.

$$\sum_{s\in\mathbb{S}} 2^{-(H(s,t)-H(t))} \leq C',$$

where C' depends only U.

Proof ▶. This follows from the Proposition on applying the Equivalence Theorem 9.1.

We can now formulate our strategy. Let us fix the string t. Suppose

$$\tau = \mu(t)$$

is the minimal program for t:

$$U(\tau) = s, \quad |\tau| = H(t).$$

We can re-write Corollary 10.1 as

$$\sum_{s\in\mathbb{S}} 2^{-(H(s,t)-H(t)+c)} \leq 1,$$

where c depends only on U. Thus the numbers

$$h_s = H(s,t) - H(t) + c$$

satisfy Kraft's inequality

$$\sum_{s\in\mathbb{S}} 2^{-h_s} \leq 1.$$

If these numbers were integers, and were recursively enumerable, then we could find a prefix-free set (depending on t)

$$P_t = \{p_{st} : s \in \mathbb{S}\}$$

such that

$$|p_{st}| \leq H(s,t) - H(t) + c.$$

Now let us prefix this set with $\mu(t) = \tau$:

$$\mu(t)P_t = \{\mu(t)p_{st} : s \in \mathbb{S}\}.$$

It is easy to see that the sets $\mu(t)P_t$ for different t are disjoint; and their union

$$P = \bigcup_{t\in\mathbb{S}} \mu(t)P_t = \{s, t \in \mathbb{S} : \mu(t)p_{st}\}.$$

is prefix-free.

Thus we may—with luck—be able to construct a machine T such that

$$T(\mu(t)p_{st}) = t$$

for each pair $s, t \in \mathbb{S}$.

From this we would deduce that

$$\begin{aligned} H(s \mid t) &\le |p_{st}| + |\langle T\rangle| \\ &\le H(s,t) - H(t) + c + |\langle T\rangle| \\ &= H(s,t) - H(t) + O(1), \end{aligned}$$

as required.

Now for the gory details. Our machine T starts by imitating U. Thus if the input string is

$$q = \tau p \quad (\tau \in \Omega(U))$$

M begines by reading in τ and computing

$$t = U(\tau).$$

However, instead of outputting t, T stores τ and t for further use.

We are only interested in the case where τ is the *minimal* program for t:

$$\tau = \mu(t), \quad |\tau| = H(t).$$

Of course, this is not generally true. But if it is not true then we do not care whether $T(q)$ is defined or not, or if it is defined what value it takes. Therefore we assume in what follows that $\tau = \mu(s)$.

By Corollary 10.1 above,

$$\sum_{s\in\mathbb{S}} 2^{-(H(s,t)-H(t)+c)} \le 1$$

Thus if we set

$$h_s = [H(s,t) - H(t) + c + 2]$$

then

$$\sum_{s\in\mathbb{S}} 2^{-h_s} \leq \frac{1}{2}.$$

As in Chapter 9, we cannot be sure that the set

$$S' = \{(s, h_s) : s \in \mathbb{S}\} \subset \mathbb{S} \times \mathbb{N}$$

is recursively enumerable. However, we can show that a slightly—but not too much—larger set S *is* recursively enumerable. (This is the reason for introducing the factor $\frac{1}{2}$ above—it allows for the difference between S and S'.)

Lemma 5. *Let*

$$S' = \{(s, h_s) : s \in \mathbb{S}\}, \quad S'' = \{(s, n) : n \geq h_s\}.$$

There exists a recursively generated set $S \subset \mathbb{S} \times \mathbb{N}$ such that

$$S' \subset S \subset S''.$$

Proof ▶. We construct an auxiliary machine M which recursively generates the set

$$\Omega(U) = \{U(p) : p \in \mathbb{S}\}.$$

As each $U(p)$ is generated, M determines if it is of the form $\langle s\rangle\langle t\rangle$ (where t is the string $U(\tau)$). If it is then M checks if the pair (s, n), where

$$n = |p| - |\tau| + c + 1$$

has already been output; if not, it outputs $\langle s\rangle\langle n\rangle$.

Since

$$|p| \geq H(s, t)$$

by definition, while by hyothesis

$$|\tau| = H(t),$$

it follows that

$$n \geq H(s, t) - H(t) + c + 1 = h_s,$$

and so $(s, n) \subset S''$. Hence

$$S \subset S''.$$

On the other hand, with the particular input $p = \mu(s, t)$,

$$|p| = H(s, t)$$

and so $n = h_s$. Thus $(s, h_t) \in S$. and so

$$S' \subset S.$$

But now

$$\sum_{(t,n)\in\mathbb{S}} 2^{-n} \leq 1,$$

since each t contributes at most

$$2^{-h_t} + 2^{-h_t-1} + 2^{-h_t-2} + \cdots = 2 \cdot 2^{-h_t}.$$

We can therefore follow Kraft's prescription for a prefix-free set. As each pair (s, n) is generated, T determines a string p_{sn} such that

$$|p_{sn}| \leq 2^{-n},$$

in such a way that the set

$$P = \{p_{sn} : (s, n) \in S\}$$

is prefix-free.

As each string p_{sn} is generated, T checks the input string q to see if

$$q = \tau p_{sn}.$$

If this is true then the computation is complete: T outputs s and halts:

$$T(\tau p_{tn}) = s.$$

One small point: in comparing τp_{sn} with the input string q, T might well go past the end of q, so that $T(q)$ is undefined. However, in that case q is certainly not of the form $\tau p_{s'n'}$, since this would imply that

$$\tau p_{s'n'} \subset \tau p_{st},$$

contradicting the 'prefix-freedom' of P.

To recap, we have constructed a machine T such that for each pair $s, t \in \mathbb{S}$ we can find a string p_{st} with

$$T(\mu(t)p_{st}) = s, \quad |p_{st}| \leq H(s, t) - H(t) + c.$$

But now, from the definition of the conditional entropy $H(s \mid t)$,

$$\begin{aligned} H(s \mid t) &\leq |p_{st}| + |\langle T \rangle| \\ &\leq H(s, t) - H(t) + c + |\langle T \rangle| \\ &= H(s, t) - H(t) + O(1). \end{aligned}$$

Summary

With the proof that

$$H(s, t) = H(t) + H(s \mid t) + O(1)$$

the basic theory of algorithmic entropy is complete.