Workshop

MIRIx people

September 9, 2014

Abstract

This is a loose compilation of notes from the MIRIx workshop at ANU on September 6–7, 2014.

Things we have shown:

We spent a lot of time talking about S, the Speed prior [Sch02]. S is a computable semi-measure. AIS is like AIξ but with S instead of ξ.

1. S is not universal, because it is computable (and there is no universal computable prior).
2. S is not a measure, for exactly the same reason that the Solomonoff prior is not a measure (some programs just print a finite string and stop).
3. However, we made up our own algorithm which does the same thing. Basically, it's clear that S is lower semi-computable. So we show that $S(x) - S(x0) - S(x1)$ is also lower semi-computable, which leads to $S(x0)$ being computable.
4. ϵ-optimal AIS is computable.
5. S effectively dominates (see Definition 1) all lower semicomputable semimeasures.
6. $S(x) \leq 1/|x|$
7. $S(x) \leq c_{S'} S'(x)$ [Sch02, Eq. 7].

Definition 1 (Effective Domination)**.** A semimeasure ν *effectively dominates* a semimeasure ρ iff there is a function $f : \mathbb{N} \to \mathbb{R}_+$ such that

$$\nu(x) \geq 2^{-f(|x|)} \rho(x).$$

Effective domination implies absolute continuity on cylinder sets (is it equivalent?).

We have that S effectively dominates all lower semicomputable semimeasures, but we are not sure whether this is useful at all. For example, if we use S for prediction of a lower semicomputable semimeasure μ, Hutter's loss bounds do not apply. Instead we get

$$KL(\mu \parallel S) \leq -\ln w_\mu = f(|x|). \tag{1}$$

In our case, $f(|x|) \to \infty$ as $|x| \to \infty$. The proof for (1) is identical to [Hut05, Thm. 3.19] except for the application of effective domination in the last step.

For deterministic μ, this gives the error bound [Hut05, Thm. 3.36]

$$E_t^{\Theta_S} \leq 2f(t),$$

where $E_t^{\Theta_S}$ is the number of errors made up to time t when using S for prediction of the sequence generated by μ. This bound is only useful if $f(t) \in o(t)$.

1 The speed prior on a monotone Turing machine

The speed prior is defined on non-monotone universal turing machines. We will show that it is straightforward to define it on a monotone turing machine, and in fact that the functions coincide.

A monotone turing machine has three situations given some input x and output y,

Definition 2 (Monotone Turing Machine [LV08, Def. 4.5.2 & Def. 4.5.3][Hut05, Def. 2.6])**.** A *monotone Turing machine* is a Turing machine with one unidirectional read-only input tape, one unidirectional write-only output tape, a finite number of work tapes, and no final states. A monotone Turing machine implements a function q that maps $x \in \mathcal{X}^\sharp$ to $y \in \mathcal{X}^\sharp$: the input tape is initialized with x and y is read from the output tape according to the following cases.

(i) $x \in \mathcal{X}^*$ is finite and $y \in \mathcal{X}^*$ is to the left of the output tape's head when the head of the input tape reads the next character right of x.

(ii) The head of the output tape writes $y \in \mathcal{X}^*$ but no more as the machine runs forever where x is infinite or the head on the read-only input tape never reads any characters right of x.

(iii) The machine writes $y \in \mathcal{X}^\omega$ to the output tape as it runs forever where x is infinite or the head on the read-only input tape never reads any characters right of x.

The speed prior is defined as follows,

$$S(x) := \sum_{i=1}^{\infty} 2^i S_i(x) where S_i(x) = \sum_{p_i \to x} 2^{-l(p)}$$

We note that $p \to x$ means our program prefix p reads p and computes output starting with x, while no prefix of p consisting of less than $l(p)$ bits outputs x.

This definition means that programs whose prefixes have already computed x do not contribute to $S(x)$ in later stages of FAST. This means that there are no more programs to sum over when we transfer to the monotone setting, since we consider a prefix-free set of programs to sum over.

Additionally, monotone Turing machines can accept strings with infinite input (see cases 2 and 3 above). However, in $S(x)$ we read in a finite prefix of the program and output a finite string as a result. We do this in stages of the algorithm FAST. This output string may be equal to x at some point in which case the program prefix will contribute to the sum, but otherwise the infinite program will never contribute to the sum.

2 Properties of S

We want to prove that Postulate 1 [Sch02] holds for S i.e. if $\mathcal{C}_t$ is the set of all strings that are computable in time t,

$$\sum_{x \notin \mathcal{C}_t} S(x) \leq \frac{1}{t} \tag{2}$$

This is straightforward,

$$\begin{aligned}
\sum_{x \notin \mathcal{C}_t} S(x) &= \sum_{x \notin \mathcal{C}_t} \sum_{i=1}^{\infty} \sum_{p \rightarrow_i x} 2^{-|p|} \\
&= \sum_{x \notin \mathcal{C}_t} \sum_{i=\log t+1}^{\infty} 2^{-i} \sum_{p \rightarrow_i x} 2^{-|p|} \\
&= 2^{-\log t} \sum_{x \notin \mathcal{C}_t} \sum_{i=1}^{\infty} 2^{-i} \sum_{p \rightarrow_{i+\log t} x} 2^{-|p|} \\
&= \frac{1}{t} \sum_{i=1}^{\infty} 2^{-i} \sum_{x \notin \mathcal{C}_t} \sum_{p \rightarrow_{i+\log t} x} 2^{-|p|} \\
&\leq \frac{1}{t} \text{(Kraft's inequality)}
\end{aligned}$$

References

[Hut05] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, 2005.

[LV08] Ming Li and Paul M. B. Vitnyi. *An Introduction to Kolmogorov Complexity and its Applications.* Texts in Computer Science. Springer, 3rd edition, 2008.

[Sch02] Jrgen Schmidhuber. The Speed Prior: a new simplicity measure yielding near-optimal computable predictions. In *Computational Learning Theory*, pages 216–228. Springer, 2002.