Universal Prediction of Selected Bits

Tor Lattimore[1], Marcus Hutter[2], and Vaibhav Gavane[3]

[1] Australian National University
tor.lattimore@anu.edu.au
[2] Australian National University and ETH Zürich
marcus.hutter@anu.edu.au
[3] VIT University, Vellore, India
vaibhav.gavane@gmail.com

Abstract. Many learning tasks can be viewed as sequence prediction problems. For example, online classification can be converted to sequence prediction with the sequence being pairs of input/target data and where the goal is to correctly predict the target data given input data and previous input/target pairs. Solomonoff induction is known to solve the general sequence prediction problem, but only if the entire sequence is sampled from a computable distribution. In the case of classification and discriminative learning though, only the targets need be structured (given the inputs). We show that the normalised version of Solomonoff induction can still be used in this case, and more generally that it can detect any recursive sub-pattern (regularity) within an otherwise completely unstructured sequence. It is also shown that the unnormalised version can fail to predict very simple recursive sub-patterns.

Keywords: Sequence prediction, Solomonoff induction, online classification, discriminative learning, algorithmic information theory.

1 Introduction

The sequence prediction problem is the task of predicting the next symbol, x_n after observing $x_1x_2\cdots x_{n-1}$. Solomonoff induction [10, 11] solves this problem by taking inspiration from Occam's razor and Epicurus' principle of multiple explanations. These ideas are formalised in the field of Kolmogorov complexity, in particular by the universal a priori semi-measure $\mathbf{M}$.

Let $\mu(x_n|x_1\cdots x_{n-1})$ be the true (unknown) probability of seeing x_n having already observed $x_1\cdots x_{n-1}$. The celebrated result of Solomonoff [10] states that if μ is computable then

$$\lim_{n\to\infty}[\mathbf{M}(x_n|x_1\cdots x_{n-1}) - \mu(x_n|x_1\cdots x_{n-1})] = 0 \text{ with } \mu\text{-probability 1} \qquad (1)$$

That is, $\mathbf{M}$ can learn the true underlying distribution from which the data is sampled with probability 1. Solomonoff induction is arguably the gold standard predictor, universally solving many (passive) prediction problems [3, 4, 10].

However, Solomonoff induction makes no guarantees if μ is not computable. This would not be problematic if it were unreasonable to predict sequences

J. Kivinen et al. (Eds.): ALT 2011, LNAI 6925, pp. 262–276, 2011.

sampled from incomputable μ, but this is not the case. Consider the sequence below, where every even bit is the same as the preceding odd bit, but where the odd bits may be chosen arbitrarily.

$$00\ 11\ 11\ 11\ 00\ 11\ 00\ 00\ 00\ 11\ 11\ 00\ 00\ 00\ 00\ 00\ 11\ 11 \tag{2}$$

Any child will quickly learn the pattern that each even bit is the same as the preceding odd bit and will correctly predict the even bits. If Solomonoff induction is to be considered a truly intelligent predictor then it too should be able to predict the even bits. More generally, it should be able to detect any computable sub-pattern. It is this question, first posed in [3, 5] and resisting attempts by experts for 6 years, that we address.

At first sight, this appears to be an esoteric question, but consider the following problem. Suppose you are given a sequence of pairs, $x_1y_1x_2y_2x_3y_3\cdots$ where x_i is the data for an image (or feature vector) of a character and y_i the corresponding ascii code (class label) for that character. The goal of online classification is to construct a predictor that correctly predicts y_i given x_i based on the previously seen training pairs. It is reasonable to assume that there is a relatively simple pattern to generate y_i given x_i (humans and computers seem to find simple patterns for character recognition). However it is not necessarily reasonable to assume there exists a simple, or even computable, underlying distribution generating the training data x_i. This problem is precisely what gave rise to discriminative learning [9].

It turns out that there exist sequences with even bits equal to preceding odd bits on which $\mathbf{M}$ fails to predict the even bits. On the other hand, it is known that $\mathbf{M}$ is a defective measure, but may be normalised to a proper measure, $\mathbf{M}_{norm}$. We show that this normalised version *does* eventually predict any recursive sub-pattern of any sequence, such as that in Equation (2). This outcome is unanticipated since (all?) other results in the field are independent of normalisation [3, 4, 8, 10]. The proofs are completely different to the standard proofs of predictive results.

2 Notation and Definitions

We use similar notation to [1, 2, 3]. For a more comprehensive introduction to Kolmogorov complexity and Solomonoff induction see [3, 4, 8, 12].

Strings. A finite binary string x is a finite sequence $x_1x_2x_3\cdots x_n$ with $x_i \in \mathcal{B} = \{0,1\}$. Its length is denoted $\ell(x)$. An infinite binary string ω is an infinite sequence $\omega_1\omega_2\omega_3\cdots$. The empty string of length zero is denoted ϵ. $\mathcal{B}^n$ is the set of all binary strings of length n. $\mathcal{B}^*$ is the set of all finite binary strings. $\mathcal{B}^\infty$ is the set of all infinite binary strings. Substrings are denoted $x_{s:t} := x_s x_{s+1}\cdots x_{t-1}x_t$ where $s,t \in \mathbb{N}$ and $s \le t$. If $s > t$ then $x_{s:t} = \epsilon$. A useful shorthand is $x_{<t} := x_{1:t-1}$. Strings may be concatenated. Let $x, y \in \mathcal{B}^*$ of length n and m respectively. Let $\omega \in \mathcal{B}^\infty$. Then,

$$xy := x_1x_2\cdots x_{n-1}x_ny_1y_2\cdots y_{m-1}y_m$$
$$x\omega := x_1x_2\cdots x_{n-1}x_n\omega_1\omega_2\omega_3\cdots$$

For $b \in \mathcal{B}$, $\neg b = 0$ if $b = 1$ and $\neg b = 1$ if $b = 0$. We write $x \sqsubseteq y$ if x is a prefix of y. Formally, $x \sqsubseteq y$ if $\ell(x) \leq \ell(y)$ and $x_i = y_i$ for all $1 \leq i \leq \ell(x)$. $x \sqsubset y$ if $x \sqsubseteq y$ and $\ell(x) < \ell(y)$.

Complexity. Here we give a brief introduction to Kolmogorov complexity and the associated notation.

Definition 1 (Inequalities). *Let f, g be real valued functions. We write $f(x) \overset{\times}{\geq} g(x)$ if there exists a constant $c > 0$ such that $f(x) \geq c \cdot g(x)$ for all x. $f(x) \overset{\times}{\leq} g(x)$ is defined similarly. $f(x) \overset{\times}{=} g(x)$ if $f(x) \overset{\times}{\leq} g(x)$ and $f(x) \overset{\times}{\geq} g(x)$.*

Definition 2 (Measures). *We call $\mu : \mathcal{B}^* \to [0,1]$ a semimeasure if $\mu(x) \geq \sum_{b \in \mathcal{B}} \mu(xb)$ for all $x \in \mathcal{B}^*$, and a probability measure if equality holds and $\mu(\epsilon) = 1$. $\mu(x)$ is the μ-probability that a sequence starts with x. $\mu(b|x) := \frac{\mu(xb)}{\mu(x)}$ is the probability of observing $b \in \mathcal{B}$ given that $x \in \mathcal{B}^*$ has already been observed. A function $P : \mathcal{B}^* \to [0,1]$ is a semi-distribution if $\sum_{x \in \mathcal{B}^*} P(x) \leq 1$ and a probability distribution if equality holds.*

Definition 3 (Enumerable Functions). *A real valued function $f : A \to \mathbb{R}$ is enumerable if there exists a computable function $f : A \times \mathbb{N} \to \mathbb{Q}$ satisfying $\lim_{t \to \infty} f(a,t) = f(a)$ and $f(a, t+1) \geq f(a,t)$ for all $a \in A$ and $t \in \mathbb{N}$.*

Definition 4 (Machines). *A Turing machine L is a recursively enumerable set (which may be finite) containing pairs of finite binary strings $(p^1, y^1), (p^2, y^2), (p^3, y^3), \cdots$.*

L is a prefix machine if the set $\{p^1, p^2, p^3 \cdots\}$ is prefix free (no program is a prefix of any other). It is a monotone machine if for all $(p,y), (q,x) \in L$ with $\ell(x) \geq \ell(y)$, $p \sqsubseteq q \implies y \sqsubseteq x$.

We define $L(p)$ to be the set of strings output by program p. This is different for monotone and prefix machines. For prefix machines, $L(p)$ contains only one element, $y \in L(p)$ if $(p,y) \in L$. For monotone machines, $y \in L(p)$ if there exists $(p,x) \in L$ with $y \sqsubseteq x$ and there does not exist a $(q,z) \in L$ with $q \sqsubset p$ and $y \sqsubseteq z$. For both machines $L(p)$ represents the output of machine L when given input p. If $L(p)$ does not exist then we say L does not halt on input p. Note that for monotone machines it is possible for the same program to output multiple strings. For example $(1,1), (1,11), (1,111), (1,1111), \cdots$ is a perfectly legitimate monotone Turing machine. For prefix machines this is not possible. Also note that if L is a monotone machine and there exists an $x \in \mathcal{B}^*$ such that $x_{1:n} \in L(p)$ and $x_{1:m} \in L(p)$ then $x_{1:r} \in L(p)$ for all $n \leq r \leq m$.

Definition 5 (Complexity). *Let L be a prefix or monotone machine then define*

$$\lambda_L(y) := \sum_{p : y \in L(p)} 2^{-\ell(p)} \qquad C_L(y) := \min_{p \in \mathcal{B}^*} \{\ell(p) : y \in L(p)\}$$

If L is a prefix machine then we write $\mathbf{m}_L(y) \equiv \lambda_L(y)$. If L is a monotone machine then we write $\mathbf{M}_L(y) \equiv \lambda_L(y)$. Note that if L is a prefix machine then

λ_L is an enumerable semi-distribution while if L is a monotone machine, λ_L is an enumerable semi-measure. In fact, every enumerable semi-measure (or semi-distribution) can be represented via some machine L as λ_L.

For prefix/monotone machine L we write L_t for the first t program/output pairs in the recursive enumeration of L, so L_t will be a finite set containing at most t pairs.[1]

The set of all monotone (or prefix) machines is itself recursively enumerable [8],[2] which allows one to define a universal monotone machine U_M as follows. Let L^i be the ith monotone machine in the recursive enumeration of monotone machines.

$$(i'p, y) \in U_M \Leftrightarrow (p, y) \in L^i$$

where i' is a prefix coding of the integer i. A universal prefix machine, denoted U_P, is defined in a similar way. For details see [8].

Theorem 6 (Universal Prefix/Monotone Machines). *For the universal monotone machine U_M and universal prefix machine U_P,*

$$\mathbf{m}_{U_P}(y) > c_L \mathbf{m}_L(y) \text{ for all } y \in \mathcal{B}^* \qquad \mathbf{M}_{U_M}(y) > c_L \mathbf{M}_L(y) \text{ for all } y \in \mathcal{B}^*$$

where $c_L > 0$ depends on L but not y.

For a proof, see [8]. As usual, we will fix reference universal prefix/monotone machines U_P, U_M and drop the subscripts by letting,

$$\mathbf{m}(y) := \mathbf{m}_{U_P}(y) \equiv \sum_{p:y\in U_P(p)} 2^{-\ell(p)} \qquad \mathbf{M}(y) := \mathbf{M}_{U_M}(y) \equiv \sum_{p:y\in U_M(p)} 2^{-\ell(p)}$$

$$K(y) := C_{U_P}(y) \equiv \min_{p\in\mathcal{B}^*} \{\ell(p) : y \in U_P(p)\} \quad Km(y) := \min_{p\in\mathcal{B}^*} \{\ell(p) : y \in U_M(p)\}$$

The choice of reference universal Turing machine is usually[3] unimportant since a different choice varies $\mathbf{m}, \mathbf{M}$ by only a multiplicative constant, while K, Km are varied by additive constants. For natural numbers n we define $K(n)$ by $K(\langle n\rangle)$ where $\langle n\rangle$ is the binary representation of n.

$\mathbf{M}$ is not a proper measure, $\mathbf{M}(x) > \mathbf{M}(x0) + \mathbf{M}(x1)$ for all $x \in \mathcal{B}^*$, which means that $\mathbf{M}(0|x) + \mathbf{M}(1|x) < 1$, so $\mathbf{M}$ assigns a non-zero probability that the sequence will end. This is because there are monotone programs p that halt, or enter infinite loops. For this reason Solomonoff introduced a normalised version, $\mathbf{M}_{norm}$ defined as follows.

[1] L_t will contain exactly t pairs unless L is finite, in which case it will contain t pairs until t is greater than the size of L. This annoyance will never be problematic.

[2] Note the enumeration may include repetition, but this is unimportant in this case.

[3] See [6] for a subtle exception. All the results in this paper are independent of universal Turing machine.

Definition 7 (Normalisation)

$$\mathbf{M}_{norm}(\epsilon) := 1 \quad \mathbf{M}_{norm}(y_n|y_{<n}) \equiv \frac{\mathbf{M}_{norm}(y_{1:n})}{\mathbf{M}_{norm}(y_{<n})} := \frac{\mathbf{M}(y_{1:n})}{\mathbf{M}(y_{<n}0) + \mathbf{M}(y_{<n}1)}.$$

This normalisation is not unique, but is philosophically and technically the most attractive and was used and defended by Solomonoff. Historically, most researchers have accepted the defective $\mathbf{M}$ for technical convenience. As mentioned, the difference seldom matters, but in this paper it is somewhat surprisingly crucial. For a discussion of normalisation, see [8].

Theorem 8. *The following are results in Kolmogorov complexity. Proofs for all can be found in [8].*

1. $\mathbf{m}(x) \stackrel{\times}{=} 2^{-K(x)}$
2. $2^{-K(xb)} \stackrel{\times}{=} 2^{-K(x\neg b)}$
3. $\mathbf{M}(x) \stackrel{\times}{\geq} \mathbf{m}(x)$
4. *If P is an enumerable semi-distribution, then* $\mathbf{m}(y) \stackrel{\times}{\geq} P(y)$
5. *If μ is an enumerable semi-measure, then* $\mathbf{M}(y) \stackrel{\times}{\geq} \mu(y)$

Note the last two results are equivalent to Theorem 6 since every enumerable semi-(measure/distribution) is generated by a monotone/prefix machine in the sense of Theorem 6 and vice-versa.

Before proceeding to our own theorems we need a recently proven result in algorithmic information theory.

Theorem 9. *[Lempp, Miller, Ng and Turetsky, 2010, unpublished, private communication]* $\lim_{n\to\infty} \frac{\mathbf{m}(\omega_{<n})}{\mathbf{M}(\omega_{<n})} = 0$*, for all* $\omega \in \mathcal{B}^\infty$.

3 $\mathbf{M}_{norm}$ Predicts Selected Bits

The following Theorem is the main positive result of this paper. It shows that any computable sub-pattern of a sequence will eventually be predicted by $\mathbf{M}_{norm}$.

Theorem 10. *Let $f : \mathcal{B}^* \to \mathcal{B} \cup \{\epsilon\}$ be a total recursive function and $\omega \in \mathcal{B}^\infty$ satisfying $f(\omega_{<n}) = \omega_n$ whenever $f(\omega_{<n}) \neq \epsilon$. If $f(\omega_{<n_i}) \neq \epsilon$ is defined for an infinite sequence $n_1, n_2, n_3, \cdots$ then* $\lim_{i\to\infty} \mathbf{M}_{norm}(\omega_{n_i}|\omega_{<n_i}) = 1$.

Essentially the Theorem is saying that if there exists a computable predictor f that correctly predicts the next bit every time it tries (i.e when $f(\omega_{<n}) \neq \epsilon$) then $\mathbf{M}_{norm}$ will eventually predict the same bits as f. By this we mean that if you constructed a predictor $f_{\mathbf{M}_{norm}}$ defined by $f_{\mathbf{M}_{norm}}(\omega_{<n}) = \arg\max_{b\in\mathcal{B}} \mathbf{M}_{norm}(b|\omega_{<n})$, then there exists an N such that $f_{\mathbf{M}_{norm}}(\omega_{<n}) = f(\omega_{<n})$ for all $n > N$ where $f(\omega_{<n}) \neq \epsilon$. For example, let f be defined by

$$f(x) = \begin{cases} x_{\ell(x)} & \text{if } \ell(x) \text{ odd} \\ \epsilon & \text{otherwise} \end{cases}$$

Now if $\omega \in \mathcal{B}^\infty$ satisfies $\omega_{2n} = f(\omega_{<2n}) = \omega_{2n-1}$ for all $n \in \mathbb{N}$ then Theorem 10 shows that $\lim_{n\to\infty} \mathbf{M}_{norm}(\omega_{2n}|\omega_{<2n}) = 1$. It says nothing about the predictive qualities of $\mathbf{M}_{norm}$ on the odd bits, on which there are no restrictions.

The proof essentially relies on using f to show that monotone programs for $\omega_{<n_i}\neg\omega_{n_i}$ can be converted to prefix programs. This is then used to show that $\mathbf{M}(\omega_{<n_i}\neg\omega_{n_i}) \stackrel{\times}{=} \mathbf{m}(\omega_{<n_i}\neg\omega_{n_i})$. The result will then follow from Theorem 9.

Theorem 10 insists that f be totally recursive and that $f(\omega_{<n}) = \epsilon$ if f refrains from predicting. One could instead allow f to be partially recursive and simply not halt to avoid making a prediction. The proof below breaks down in this case and we suspect that Theorem 10 will become invalid if f is permitted to be only partially recursive.

Proof (Theorem 10). We construct a machine L from U_M consisting of all programs that produce output that f would not predict. We then show that these programs essentially form a prefix machine. Define L by the following process

1. $L := \emptyset$ and $t := 1$.
2. Let (p, y) be the tth pair in U_M.
3. Let i be the smallest natural number such that $y_i \neq f(y_{<i}) \neq \epsilon$. That is, i is the position at which f makes its first mistake when predicting y. If f makes no prediction errors then i doesn't exist.[4]
4. If i exists then $L := L \cup \{(p, y_{1:i})\}$ (Note that we do not allow L to contain duplicates).
5. $t := t + 1$ and go to step 2.

Since f is totally recursive and U_M is recursively enumerable, the process above shows that L is recursively enumerable. It is easy to see that L is a monotone machine. Further, if $(p, y), (q, x) \in L$ with $p \sqsubseteq q$ then $y = x$. This follows since by monotonicity we would have that $y \sqsubseteq x$, but $f(x_{<\ell(y)}) = f(y_{<\ell(y)}) \neq y_{\ell(y)} = x_{\ell(y)}$ and by steps 3 and 4 in the process above we have that $\ell(x) = \ell(y)$.

Recall that L_t is the tth enumeration of L and contains t elements. Define $\bar{L}_t \subseteq L_t$ to be the largest prefix free set of shortest programs. Formally, $(p, y) \in \bar{L}_t$ if there does not exist a $(q, x) \in L_t$ such that $q \sqsubset p$. For example, if $L_t = (1, 001), (11, 001), (01, 11110), (010, 11110)$ then $\bar{L}_t = (1, 001), (01, 11110)$. If we now added $(0, 11110)$ to L_t to construct L_{t+1} then $\bar{L}_{t+1}$ would be $(1, 001), (0, 11110)$.

Since L_t is finite, $\bar{L}_t$ is easily computable from L_t. Therefore the following function is computable.

$$P(y, t) := \sum_{p:(p,y)\in\bar{L}_t} 2^{-\ell(p)} \geq 0.$$

[4] This is where the problem lies for partially recursive prediction functions. Computing the smallest i for which f predicts incorrectly is incomputable if f is only partially recursive, but computable if it is totally recursive. It is this distinction that allows L to be recursively enumerable, and so be a machine.

Now $\bar{L}_t$ is prefix free, so by Kraft's inequality $\sum_{y\in\mathcal{B}^*} P(y,t) \leq 1$ for all $t \in \mathbb{N}$. We now show that $P(y,t+1) \geq P(y,t)$ for all $y \in \mathcal{B}^*$ and $t \in \mathbb{N}$ which proves that $P(y) = \lim_{t\to\infty} P(y,t)$ exists and is a semi-distribution.

Let (p,y) be the program/output pair in L_{t+1} but not in L_t. To see how $P(\cdot,t)$ compares to $P(\cdot,t+1)$ we need to compare $\bar{L}_t$ and $\bar{L}_{t+1}$. There are three cases:

1. There exists a $(q,x) \in L_t$ with $q \sqsubset p$. In this case $\bar{L}_{t+1} = \bar{L}_t$.
2. There does not exist a $(q,x) \in L_t$ such that $p \sqsubset q$. In this case (p,y) is simply added to $\bar{L}_t$ to get $\bar{L}_{t+1}$ and so $\bar{L}_t \subset \bar{L}_{t+1}$. Therefore $P(\cdot,t+1) \geq P(\cdot,t)$ is clear.
3. There does exist a $(q,x) \in \bar{L}_t$ such that $p \sqsubset q$. In this case $\bar{L}_{t+1}$ differs from $\bar{L}_t$ in that it contains (p,y) but not (q,x). Since $p \sqsubset q$ we have that $y = x$. Therefore $P(y,t+1) - P(y,t) = 2^{-\ell(p)} - 2^{-\ell(q)} > 0$ since $p \sqsubset q$. For other values, $P(\cdot,t) = P(\cdot,t+1)$.

Note that it is not possible that $p = q$ since then $x = y$ and duplicates are not added to L. Therefore P is an enumerable semi-distribution. By Theorem 8 we have

$$\mathbf{m}(\omega_{<n_i}\neg\omega_{n_i}) \overset{\times}{\geq} P(\omega_{<n_i}\neg\omega_{n_i}) \tag{3}$$

where the constant multiplicative fudge factor in the $\overset{\times}{\geq}$ is independent of i. Suppose $\omega_{<n_i}\neg\omega_{n_i} \in U_M(p)$. Therefore there exists a y such that $\omega_{<n_i}\neg\omega_{n_i} \sqsubseteq y$ and $(p,y) \in U_M$. By parts 2 and 3 of the process above, $(p,\omega_{<n_i}\neg\omega_{n_i})$ is added to L. Therefore there exists a $T \in \mathbb{N}$ such that $(p,\omega_{<n_i}\neg\omega_{n_i}) \in L_t$ for all $t \geq T$.

Since $\omega_{<n_i}\neg\omega_{n_i} \in U_M(p)$, there does not exist a $q \sqsubset p$ with $\omega_{<n_i}\neg\omega_{n_i} \in U_M(q)$. Therefore eventually, $(p,\omega_{<n_i}\neg\omega_{n_i}) \in \bar{L}_t$ for all $t \geq T$. Since every program in U_M for $\omega_{<n_i}\neg\omega_{n_i}$ is also a program in L, we get

$$\lim_{t\to\infty} P(\omega_{<n_i}\neg\omega_{n_i},t) \equiv P(\omega_{<n_i}\neg\omega_{n_i}) = \mathbf{M}(\omega_{<n_i}\neg\omega_{n_i}).$$

Next,

$$\mathbf{M}_{norm}(\neg\omega_{n_i}|\omega_{<n_i}) \equiv \frac{\mathbf{M}(\omega_{<n_i}\neg\omega_{n_i})}{\mathbf{M}(\omega_{<n_i}\omega_{n_i}) + \mathbf{M}(\omega_{<n_i}\neg\omega_{n_i})} \tag{4}$$

$$\overset{\times}{\leq} \frac{\mathbf{m}(\omega_{<n_i}\neg\omega_{n_i})}{\mathbf{M}(\omega_{1:n_i})} \tag{5}$$

$$\overset{\times}{=} \frac{\mathbf{m}(\omega_{1:n_i})}{\mathbf{M}(\omega_{1:n_i})} \tag{6}$$

where Equation (4) follows by the definition of $\mathbf{M}_{norm}$. Equation (5) follows from Equation (3) and algebra. Equation (6) follows since $\mathbf{m}(xb) \overset{\times}{=} 2^{-K(xb)} \overset{\times}{=} 2^{-K(x\neg b)} \overset{\times}{=} \mathbf{m}(x\neg b)$, which is Theorem 8. However, by Theorem 9, $\lim_{i\to\infty} \frac{\mathbf{m}(\omega_{<n_i})}{\mathbf{M}(\omega_{<n_i})} = 0$ and so $\lim_{i\to\infty} \mathbf{M}_{norm}(\neg\omega_{n_i}|\omega_{<n_i}) = 0$. Therefore $\lim_{i\to\infty} \mathbf{M}_{norm}(\omega_{n_i}|\omega_{<n_i}) = 1$ as required. □

We have remarked already that Theorem 10 is likely not valid if f is permitted to be a partial recursive function that only output on sequences for which they make a prediction. However, there is a class of predictors larger than the totally recursive ones of Theorem 10, which $\mathbf{M}_{norm}$ still learns.

Theorem 11. *Let $f : \mathcal{B}^* \to \mathcal{B} \cup \{\epsilon\}$ be a partial recursive function and $\omega \in \mathcal{B}^\infty$ satisfying*

1. *$f(\omega_{<n})$ is defined for all n.*
2. *$f(\omega_{<n}) = \omega_n$ whenever $f(\omega_{<n}) \neq \epsilon$.*

If $f(\omega_{<n_i}) \in \mathcal{B}$ for an infinite sequence $n_1, n_2, n_3, \cdots$ then

$$\lim_{i\to\infty} \mathbf{M}_{norm}(\omega_{n_i}|\omega_{<n_i}) = 1.$$

The difference between this result and Theorem 10 is that f need only be defined on all prefixes of at least one $\omega \in \mathcal{B}^\infty$ and not everywhere in $\mathcal{B}^*$. This allows for a slightly broader class of predictors. For example, let $\omega = p^1 b^1 p^2 b^2 p^3 b^3 \cdots$ where p^i is some prefix machine that outputs at least one bit and b^i is the first bit of that output. Now there exists a computable f such that $f(p^1 b^1 \cdots p^{i-1} b^{i-1} p^i) = b^i$ for all i and $f(\omega_{<n}) = \epsilon$ whenever $\omega_n \neq b^i$ for some i (f only tries to predict the outputs). By Theorem 11, $\mathbf{M}_{norm}$ will correctly predict b^i.

The proof of Theorem 11 is almost identical to that of Theorem 10, but with one additional subtlety.

Proof sketch. The proof follows that of Theorem 10 until the construction of L. This breaks down because step 3 is no longer computable since f may not halt on some string that is not a prefix of ω. The modification is to run steps 2-4 in parallel for all t and only adding $(p, y_{1:i})$ to L once it has been proven that $f(y_{<i}) \neq y_i$ and $f(y_{<k})$ halts for all $k < i$, and either chooses not to predict (outputs ϵ), or predicts correctly. Since f halts on all prefixes of ω, this does not change L for any programs we care about and the remainder of the proof goes through identically. □

It should be noted that this new class of predictors is still less general than allowing f to an arbitrary partial recursive predictor. For example, a partial recursive f can predict the ones of the halting sequence, while choosing not to predict the zeros (the non-halting programs). It is clear this cannot be modified into a computable f predicting both ones and zeros, or predicting ones and outputting ϵ rather than zero, as this would solve the halting problem.

4 M Fails to Predict Selected Bits

The following theorem is the corresponding negative result that while $\mathbf{M}_{norm}$ always predicts recursive sub-patterns, $\mathbf{M}$ can fail to do so. This essentially means that the key problem with $\mathbf{M}$ is that it is a defective measure.

Theorem 12. *Let $f : \mathcal{B}^* \to \mathcal{B} \cup \{\epsilon\}$ be the total recursive function defined by,*

$$f(z) := \begin{cases} z_{\ell(z)} & \text{if } \ell(z) \text{ odd} \\ \epsilon & \text{otherwise} \end{cases}$$

There exists an infinite string $\omega \in \mathcal{B}^\infty$ *with* $\omega_{2n} = f(\omega_{<2n}) \equiv \omega_{2n-1}$ *for all* $n \in \mathbb{N}$ *such that*

$$\liminf_{n\to\infty} \mathbf{M}(\omega_{2n}|\omega_{<2n}) < 1.$$

The proof requires some lemmas.

Lemma 13. $\mathbf{M}(xy)$ *can be bounded as follows.*

$$2^{K(\ell(x))}\mathbf{M}(y) \overset{\times}{\geq} \mathbf{M}(xy) \overset{\times}{\geq} \mathbf{M}(y)2^{-K(x)}. \tag{7}$$

Proof. Both inequalities are proven relatively easily by normal methods as used in [8] and elsewhere. Nevertheless we present them as a warm-up to the slightly more subtle proof later.

Now construct monotone machine L, which we should think of as taking two programs as input. The first, a prefix program p, the output of which we view as a natural number n. The second, a monotone program. We then simulate the monotone machine and strip the first n bits of its output. L is formally defined as follows.

1. $L := \emptyset$, $t := 1$
2. Let $(p, n), (q, y)$ be the tth pair of program/outputs in $U_P \times U_M$, which is enumerable.
3. If $\ell(y) \geq n$ then add $(pq, y_{n+1:\ell(y)})$ to L
4. $t := t + 1$ and go to step 2

By construction, L is enumerable and is a monotone machine. Note that if $xy \in U_M(q)$ and $\ell(x) \in U_P(p)$ then $y \in L(pq)$. Now,

$$\mathbf{M}(y) \overset{\times}{\geq} \mathbf{M}_L(y) \equiv \sum_{r:y\in L(r)} 2^{-\ell(r)} \geq \sum_{q,p:xy\in U_M(q),\ell(x)\in U_P(p)} 2^{-\ell(pq)} \tag{8}$$

$$= \sum_{q:xy\in U_M(q)} 2^{-\ell(q)} \sum_{p:\ell(x)\in U_P(p)} 2^{-\ell(p)} \equiv \mathbf{M}(xy)\mathbf{m}(\ell(x)) \tag{9}$$

$$\overset{\times}{=} \mathbf{M}(xy)2^{-K(\ell(x))} \tag{10}$$

where Equation (8) follows by Theorem 6, definitions and because if $xy \in U_M(q)$ and $\ell(x) \in U_P(p)$ then $y \in L(pq)$. Equation (9) by algebra, definitions. Equation (10) by Theorem 8.

The second inequality is proved similarly. We define a machine L as follows,

1. $L = \emptyset$, $t := 1$
2. Let $(q, x), (r, y)$ be the tth element in $U_P \times U_M$, which is enumerable.
3. Add (qr, xy) to L
4. $t := t + 1$ and go to step 2

It is easy to show that L is monotone by using the properties of U_P and U_M. Now,

$$\mathbf{M}(xy) \stackrel{\times}{\geq} \mathbf{M}_L(xy) \equiv \sum_{p:xy\in L(p)} 2^{-\ell(p)} \geq \sum_{q,r:x\in U_P(q),y\in U_M(r)} 2^{-\ell(qr)}$$
$$= \sum_{q:x\in U_P(q)} 2^{-\ell(q)} \sum_{r:y\in U_M(r)} 2^{-\ell(r)} \equiv \mathbf{m}(x)\mathbf{M}(y) \stackrel{\times}{=} 2^{-K(x)}\mathbf{M}(y).$$

□

Lemma 14. *There exists an $\omega \in \mathcal{B}^\infty$ such that*

$$\liminf_{n\to\infty} [\mathbf{M}(0|\omega_{<n}) + \mathbf{M}(1|\omega_{<n})] = 0.$$

Proof. First we show that for each $\delta > 0$ there exists a $z \in \mathcal{B}^*$ such that $\mathbf{M}(0|z) + \mathbf{M}(1|z) < \delta$. This result is already known and is left as an exercise (4.5.6) with a proof sketch in [8]. For completeness, we include a proof. Recall that $\mathbf{M}(\cdot, t)$ is the function approximating $\mathbf{M}(\cdot)$ from below. Fixing an n, define $z \in \mathcal{B}^*$ inductively as follows.

1. $z := \epsilon$
2. Let t be the first natural number such that $\mathbf{M}(zb, t) > 2^{-n}$ for some $b \in \mathcal{B}$.
3. If t exists then $z := z\neg b$ and repeat step 2. If t does not exist then z is left unchanged (forever).

Note that z must be finite since each time it is extended, $\mathbf{M}(zb, t) > 2^{-n}$. Therefore $\mathbf{M}(z\neg b, t) < \mathbf{M}(z, t) - 2^{-n}$ and so each time z is extended, the value of $\mathbf{M}(z, t)$ decreases by at least 2^{-n} so eventually $\mathbf{M}(zb, t) < 2^{-n}$ for all $b \in \mathcal{B}$. Now once the z is no longer being extended (t does *not* exist in step 3 above) we have

$$\mathbf{M}(z0) + \mathbf{M}(z1) \leq 2^{1-n}. \tag{11}$$

However we can also show that $\mathbf{M}(z) \stackrel{\times}{\geq} 2^{-K(n)}$. The intuitive idea is that the process above requires only the value of n, which can be encoded in $K(n)$ bits. More formally, let p be such that $n \in U_P(p)$ and note that the following set is recursively enumerable (but not recursive) by the process above.

$$L_p := (p, \epsilon), (p, z_{1:1}), (p, z_{1:2}), (p, z_{1:3}), \cdots, (p, z_{1:\ell(z)-1}), (p, z_{1:\ell(z)}).$$

Now take the union of all such sets, which is a) recursively enumerable since U_P is, and b) a monotone machine because U_P is a prefix machine.

$$L := \bigcup_{(p,n)\in U_P} L_p.$$

Therefore

$$\mathbf{M}(z) \stackrel{\times}{\geq} \mathbf{M}_L(z) \geq 2^{-K(n)} \tag{12}$$

where the first inequality is from Theorem 6 and the second follows since if n^* is the program of length $K(n)$ with $U_P(n^*) = n$ then $(n^*, z_{1:\ell(z)}) \in L$. Combining Equations (11) and (12) gives

$$\mathbf{M}(0|z) + \mathbf{M}(1|z) \overset{\times}{\leq} 2^{1-n+K(n)}.$$

Since this tends to zero as n goes to infinity,[5] for each $\delta > 0$ we can construct a $z \in \mathcal{B}^*$ satisfying $\mathbf{M}(0|z) + \mathbf{M}(1|z) < \delta$, as required. For the second part of the proof, we construct ω by concatenation.

$$\omega := z^1 z^2 z^3 \cdots$$

where $z^n \in \mathcal{B}^*$ is chosen such that,

$$\mathbf{M}(0|z^n) + \mathbf{M}(1|z^n) < \delta_n \tag{13}$$

with δ_n to be chosen later. Now,

$$\mathbf{M}(b|z^1 \cdots z^n) \equiv \frac{\mathbf{M}(z^1 \cdots z^n b)}{\mathbf{M}(z^1 \cdots z^n)} \tag{14}$$

$$\overset{\times}{\leq} \left[2^{K(\ell(z^1 \cdots z^{n-1}))+K(z^1 \cdots z^{n-1})}\right] \frac{\mathbf{M}(z^n b)}{\mathbf{M}(z^n)} \tag{15}$$

$$\equiv \left[2^{K(\ell(z^1 \cdots z^{n-1}))+K(z^1 \cdots z^{n-1})}\right] \mathbf{M}(b|z^n) \tag{16}$$

where Equation (14) is the definition of conditional probability. Equation (15) follows by applying Lemma 13 with $x = z^1 z^2 \cdots z^{n-1}$ and $y = z^n$ or $z^n b$. Equation (16) is again the definition of conditional probability. Now let

$$\delta_n = \frac{2^{-n}}{2^{K(\ell(z^1 \cdots z^{n-1}))+K(z^1 \cdots z^{n-1})}}.$$

Combining this with Equations (13) and (16) gives

$$\mathbf{M}(0|z^1 \cdots z^n) + \mathbf{M}(1|z^1 \cdots z^n) \overset{\times}{\leq} 2^{-n}.$$

Therefore,

$$\liminf_{n \to \infty} \left[\mathbf{M}(0|\omega_{<n}) + \mathbf{M}(1|\omega_{<n})\right] = 0$$

as required. □

Proof (Theorem 12). Let $\bar{\omega} \in \mathcal{B}^\infty$ be defined by $\bar{\omega}_{2n} := \bar{\omega}_{2n-1} := \omega_n$ where ω is the string defined in the previous lemma. Recall $U_M := \left\{(p^1, y^1), (p^2, y^2), \cdots\right\}$ is the universal monotone machine. Define monotone machine L by the following process,

[5] An integer n can easily be encoded in $2 \log n$ bits, so $K(n) \leq 2 \log n + c$ for some $c > 0$ independent of n.

1. $L = \emptyset$, $t = 1$
2. Let (p, y) be the tth element in the enumeration of U_M
3. Add $(p, y_1 y_3 y_5 y_7 \cdots)$ to L
4. $t := t + 1$ and go to step 2.

Therefore if $\bar{\omega}_{<2n} \in U_M(p)$ then $\omega_{1:n} \in L(p)$. By identical reasoning as elsewhere,

$$\mathbf{M}(\omega_{1:n}) \stackrel{\times}{\geq} \mathbf{M}(\bar{\omega}_{<2n}). \tag{17}$$

In fact, $\mathbf{M}(\omega_{1:n}) \stackrel{\times}{=} \mathbf{M}(\bar{\omega}_{<2n})$, but this is unnecessary. Let $P := \{p : \exists b \in \mathcal{B} \text{ s.t } \omega_{1:n} b \in U_M(p)\}$ and $Q := \{p : \omega_{1:n} \in U_M(p)\} \supset P$. Therefore

$$\begin{aligned} 1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n}) &= 1 - \frac{\sum_{p \in P} 2^{-\ell(p)}}{\sum_{q \in Q} 2^{-\ell(q)}} \\ &= \frac{\sum_{p \in Q-P} 2^{-\ell(p)}}{\sum_{q \in Q} 2^{-\ell(q)}}. \end{aligned}$$

Now let $\bar{P} := \{p : \exists b \in \mathcal{B} \text{ s.t } \bar{\omega}_{<2n} b \in U_M(p)\}$ and $\bar{Q} := \{p : \bar{\omega}_{<2n} \in U_M(p)\} \supset \bar{P}$. Define monotone machine L by the following process

1. $L = \emptyset$, $t := 1$
2. Let (p, y) be the tth program/output pair in U_M
3. Add $(p, y_1 y_1 y_2 y_2 \cdots y_{\ell(y)-1} y_{\ell(y)-1} y_{\ell(y)})$ to L
4. $t := t + 1$ and go to step 2.

Let $p \in Q - P$. Therefore $\omega_{1:n} \in U_M(p)$ and $\omega_{1:n} b \notin U_M(p)$ for any $b \in \mathcal{B}$. Therefore $\bar{\omega}_{<2n} \in L(p)$ while $\bar{\omega}_{<2n} b \notin L(p)$ for any $b \in \mathcal{B}$. Now there exists an i such that L is the ith machine in the enumeration of monotone machines, L^i.

Therefore, by the definition of the universal monotone machine U_M we have that $\bar{\omega}_{<2n} b \notin U_M(i'p) = L^i(p) = L(p) \ni \bar{\omega}_{<2n}$ and $U_M(i'p) = L(p)$ for any $b \in \mathcal{B}$. Therefore $i'p \in \bar{Q} - \bar{P}$ and so,

$$\sum_{q \in \bar{Q} - \bar{P}} 2^{-\ell(q)} \geq \sum_{p : i'p \in \bar{Q} - \bar{P}} 2^{-\ell(i'p)} \geq \sum_{p \in Q - P} 2^{-\ell(i'p)} \stackrel{\times}{=} \sum_{p \in Q - P} 2^{-\ell(p)}. \tag{18}$$

Therefore

$$1 - \mathbf{M}(0|\bar{\omega}_{<2n}) - \mathbf{M}(1|\bar{\omega}_{<2n}) \equiv \frac{\sum_{p \in \bar{Q} - \bar{P}} 2^{-\ell(p)}}{\mathbf{M}(\bar{\omega}_{<2n})} \tag{19}$$

$$\stackrel{\times}{\geq} \frac{\sum_{p \in Q - P} 2^{-\ell(p)}}{\mathbf{M}(\omega_{1:n})} \tag{20}$$

$$\equiv 1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n}) \tag{21}$$

where Equation (19) follows from the definition of $\bar{P}$, $\bar{Q}$ and $\mathbf{M}$. Equation (20) by (18) and (17). Equation (21) by the definition of P, Q and $\mathbf{M}$. Therefore by Lemma 14 we have

$$\limsup_{n \to \infty} [1 - \mathbf{M}(0|\bar{\omega}_{<2n}) - \mathbf{M}(1|\bar{\omega}_{<2n})] \stackrel{\times}{\geq} \limsup_{n \to \infty} [1 - \mathbf{M}(0|\omega_{1:n}) - \mathbf{M}(1|\omega_{1:n})] = 1.$$

Therefore

$$\liminf_{n\to\infty} \mathbf{M}(\bar{\omega}_{2n}|\bar{\omega}_{<2n}) < 1$$

as required. □

Note that $\lim_{n\to\infty} \mathbf{M}(\bar{\omega}_{2n}|\bar{\omega}_{<2n}) \neq 0$ in fact, one can show that there exists a $c > 0$ such that $\mathbf{M}(\bar{\omega}_{2n}|\bar{\omega}_{<2n}) > c$ for all $n \in \mathbb{N}$.

5 Discussion

Summary. Theorem 10 shows that if an infinite sequence contains a computable sub-pattern then the normalised universal semi-measure $\mathbf{M}_{norm}$ will eventually predict it. This means that Solomonoff's normalised version of induction is effective in the classification example given in the introduction. Note that we have only proven the binary case, but expect the proof will go through identically for arbitrary finite alphabet.

On the other hand, Theorem 12 shows that plain $\mathbf{M}$ can fail to predict such structure and that it does so because it is not a proper measure. These results are surprising since (all?) other predictive results, including Equation (1) and many others in [3, 4, 8, 10], do not rely on normalisation.

Consequences. We have shown that $\mathbf{M}$ really is deficient on some sequences that may still be of interest. This forces us to use $\mathbf{M}_{norm}$ which has one major drawback, it is not enumerable. Since $\mathbf{M}$ is enumerable, we have some hope of approximating it using standard compression algorithms or more brute force methods. However $\mathbf{M}_{norm}$ is only approximable,[6] which makes it substantially more challenging or impossible to approximate. This disadvantage is not as great as it seems since the predictive distribution $\mathbf{M}(b|x)$ is also only approximable.

Open Questions. A number of open questions were encountered in writing this paper.

1. Extend Theorem 10 to the stochastic case where a sub-pattern is generated stochastically from a computable distribution rather than merely a computable function. It seems likely that a different approach will be required to solve this problem.
2. Another interesting question is to strengthen the result by proving a convergence rate. It may be possible to prove that under the same conditions as Theorem 10 that $\sum_{i=1}^{\infty} [1 - \mathbf{M}_{norm}(\omega_{n_i}|\omega_{<n_i})] \overset{\times}{\leq} K(f)$ where $K(f)$ is the (prefix) complexity of the predicting function f. Again, if this is even possible, it will likely require a different approach.
3. Prove or disprove the validity of Theorem 10 when the totally recursive prediction function f (or the modified predictor of Theorem 11) is replaced by a partially recursive function.

[6] A function f is approximable if there exists a computable function $f(\cdot, t)$ with $\lim_{t\to\infty} f(\cdot, t) = f(\cdot)$. Convergence need not be monotonic.

Acknowledgements. We thank Wen Shao and reviewers for valuable feedback on earlier drafts and the Australian Research Council for support under grant DP0988049.

References

[1] Gács, P.: On the relation between descriptional complexity and algorithmic probability. Theoretical Computer Science 22(1-2), 71–93 (1983)
[2] Gács, P.: Expanded and improved proof of the relation between description complexity and algorithmic probability (2008) (unpublished)
[3] Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004)
[4] Hutter, M.: On universal prediction and Bayesian confirmation. Theoretical Computer Science 384(1), 33–48 (2007)
[5] Hutter, M.: Open problems in universal induction & intelligence. Algorithms 3(2), 879–906 (2009)
[6] Hutter, M., Muchnik, A.A.: On semimeasures predicting Martin-Löf random sequences. Theoretical Computer Science 382(3), 247–261 (2007)
[7] Lempp, S., Miller, J., Ng, S., Turetsky, D.: Complexity inequality. Unpublished, private communication (2010)
[8] Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer, Heidelberg (2008)
[9] Long, P., Servedio, R.: Discriminative Learning Can Succeed Where Generative Learning Fails. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 319–334. Springer, Heidelberg (2006)
[10] Solomonoff, R.: A formal theory of inductive inference, Part I. Information and Control 7(1), 1–22 (1964)
[11] Solomonoff, R.: A formal theory of inductive inference, Part II. Information and Control 7(2), 224–254 (1964)
[12] Zvonkin, A.K., Levin, L.A.: The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. Russian Mathematical Surveys 25(6), 83 (1970)

Table of Notation

Symbol	Description
$\mathcal{B}$	Binary symbols, 0 and 1
$\mathbb{Q}$	Rational numbers
$\mathbb{N}$	Natural numbers
$\mathcal{B}^*$	The set of all finite binary strings
$\mathcal{B}^\infty$	The set of all infinite binary strings
x, y, z	Finite binary strings
ω	An infinite binary string
$\bar{\omega}$	An infinite binary string with even bits equal to preceding odd bits
$\ell(x)$	The length of binary string x
$\neg b$	The negation of binary symbol b. $\neg b = 0$ if $b = 1$ and $\neg b = 1$ if $b = 0$
p, q	Programs
μ	An enumerable semi-measure
$\mathbf{M}$	The universal enumerable semi-measure
$\mathbf{M}_{norm}$	The normalised version of the universal enumerable semi-measure
$\mathbf{m}$	The universal enumerable semi-distribution
$K(f)$	The prefix Kolmogorov complexity of a function f
L	An enumeration of program/output pairs defining a machine
U_M	The universal monotone machine
U_P	The universal prefix machine
$\overset{\times}{\geq}$	$f(x) \overset{\times}{\geq} g(x)$ if there exists a $c > 0$ such that $f(x) > c \cdot g(x)$ for all x
$\overset{\times}{\leq}$	$f(x) \overset{\times}{\leq} g(x)$ if there exists a $c > 0$ such that $f(x) < c \cdot g(x)$ for all x
$\overset{\times}{=}$	$f(x) \overset{\times}{=} g(x)$ if $f(x) \overset{\times}{\geq} g(x)$ and $f(x) \overset{\times}{\leq} g(x)$
$x \sqsubset y$	x is a prefix of y and $\ell(x) < \ell(y)$
$x \sqsubseteq y$	x is a prefix of y