Is there an Elegant Universal Theory of Prediction?

Shane Legg

Dalle Molle Institute for Artificial Intelligence
Manno-Lugano
Switzerland

17th International Conference on
Algorithmic Learning Theory

Is there an Elegant Universal Theory of Prediction?

Solomonoff's *incomputable* model of induction rapidly learns to make optimal predictions for *any* computable sequence, including probabilistic ones.

Lempel-Ziv, Context Tree Weighting and other *computable* predictors can predict *some* computable sequences.

Question

Does there exist an elegant computable predictor that is in some sense universal, or at least universal over large sets of simple sequences?

Basic notation

$\mathbb{B} := \{0, 1\}$

$\mathbb{B}^*$:= the set of binary *strings*

$\mathbb{B}^\infty$:= the set of infinite binary *sequences*

x_n := the n^{th} symbol in the string $x \in \mathbb{B}^*$

$x_{i:j}$:= the substring $x_i x_{i+1} \dots x_{j-1} x_j$

$|x|$:= the length of the string x

$f(x) \stackrel{+}{<} g(x) \ :\Leftrightarrow \ f(x) < g(x) + c$ for some independent constant c

Kolmogorov complexity

In this work we will use Kolmogorov complexity to measure the complexity of both sequences and strings.

For a sequence $\omega \in \mathbb{B}^\infty$ and universal Turing machine $\mathcal{U}$,

$$K(\omega) := \min_{p \in \mathbb{B}^*} \{|p| : \mathcal{U}(p) = \omega\}$$

In words: The *Kolmogorov complexity* of a sequence ω is the length of the shortest program that genereates ω.

For strings, $\mathcal{U}(p)$ must halt with output x.

Basic definitions

Definition

A sequence $\omega \in \mathbb{B}^\infty$ is a **computable binary sequence** if there exists a program $q \in \mathbb{B}^*$ that writes ω to a one-way output tape when run on a monotone universal Turing machine $\mathcal{U}$. We denote the set of all computable sequences by $\mathcal{C}$.

Definition

A **computable binary predictor** is a program $p \in \mathbb{B}^*$ that computes a total function $\mathbb{B}^* \to \mathbb{B}$.

Ideally a predictor's output should be the next symbol in the sequence, that is, $p(x_{1:n}) = x_{n+1}$.

Predictability in the limit

We will place no limits on the predictor's

- computation time
- storage capacity

Furthermore we will only consider predictability in the limit:

Definition

We say that a predictor p can **learn to predict** a sequence $\omega := x_1 x_2 \ldots \in \mathbb{B}^\infty$ if there exists $m \in \mathbb{N}$ such that $\forall n \geq m : p(x_{1:n}) = x_{n+1}$.

Sets of predictors

We will be focused on the set of all predictors that are able to predict some specific sequence ω, or all sequences in some set of sequences S.

Definition

Let $P(\omega)$ be the set of all predictors able to learn to predict ω.
For a set of sequences S, let $P(S) := \bigcap_{\omega \in S} P(\omega)$.

Every computable sequence can be predicted

Lemma

$\forall \omega \in \mathcal{C}, \exists p \in P(\omega) : K(p) \stackrel{+}{<} K(\omega).$

In words: Every computable sequence can be predicted by at least one predictor. This predictor need not be significantly more complex than the sequence.

Proof sketch: Take the program q that generates the sequence ω and convert this into a "predictor" p that always outputs the $(n+1)^{th}$ symbol of ω for *any* input $x_{1:n}$. Clearly p is not significantly more complex than q and correctly "predicts" ω (and only ω!)

Simple predictors for complex strings

Lemma

There exists a predictor p such that $\forall n \in \mathbb{N}, \exists \omega \in \mathcal{C} : p \in P(\omega)$ and $K(\omega) > n$.

In words: There exists a predictor that is able to learn to predict *some* strings of arbitrarily high complexity.

Proof sketch: A predictor that always predicts 0 can predict any sequence ω of the form $x0^*$ no matter how complex $x \in \mathbb{B}^*$ is.

In a sense ω becomes a simple sequence once it has converged to 0. This is not necessary: Consider a prediction program that detects when the input sequence is a repeating string and then predicts accordingly. Clearly, some sequences with arbitrarily high "tail complexity" can also be predicted by simple predictors.

There is no universal computable predictor

Lemma

For any predictor p there exists a computable sequence $\omega := x_1 x_2 \ldots \in \mathcal{C}$ such that $\forall n \in \mathbb{N} : p(x_{1:n}) \neq x_{n+1}$ and $K(\omega) \stackrel{+}{<} K(p)$.

In words: For every computable predictor there exists a computable sequence which it cannot predict at all, furthermore this sequence doesn't have to be significantly more complex than the predictor.

Proof sketch: For any prediction program p we can construct a sequence generation program q that always outputs the opposite of what p predicts given the sequence so far. Clearly p will always mis-predict this sequence and q is not much more complex than p.

Prediction of simple computable sequences

As there is no universal computable sequence predictor, a weaker goal is to be able to predict all "simple" computable sequences.

Definition

For $n \in \mathbb{N}$, let $\mathcal{C}_n := \{\omega \in \mathcal{C} : K(\omega) \leq n\}$.

Definition

Let $P_n := P(\mathcal{C}_n)$ be the set of predictors able to learn to predict all sequences in $\mathcal{C}_n$.

A key question then is whether $P_n \neq \varnothing$ for large n. That is, whether powerful predictors exist that can predict all sequences up to a high level of complexity.

Predictors for sets of bounded complexity sequences exist

Lemma

$\forall n \in \mathbb{N}, \exists p \in P_n : K(p) \stackrel{+}{<} n + O(\log_2 n).$

In words: Prediction algorithms exist that can learn to predict all sequences up to a given complexity, and these predictors need not be significantly more complex than the sequences they can predict.

Proof sketch: Let h be the number of valid sequence generation programs of length up to n bits. Construct a predictor that on input $x_{1:k}$ simulates all programs of length up to n bits until h of these produce sequences of length $n+1$. In the limit only the h programs that are valid sequence generators will do this. Finally, predict according to the lexicographically first program that is consistent with the input string $x_{1:k}$. As h can be encoded in $n + O(\log_2 n)$ bits the result follows.

Can we do better?

Do there exist simple predictors that can predict all sequences up to a high level of complexity?

Theorem

$\forall n \in \mathbb{N} : p \in P_n \Rightarrow K(p) \stackrel{+}{>} n.$

In words: If a predictor can predict all sequences up to a complexity of n then the complexity of the predictor must be at least n.

Proof sketch: Follows immediately from the previous result that a simple predictor must fail for some equally simple sequence.

Note: This result is true for any measure of complexity for which the inversion of a single bit is an inexpensive operation.

Complexity of prediction

The previous results suggest the following definition,

Definition

$$\dot{K}(\omega) := \min_{p \in \mathbb{B}^*} \{|p| : p \in P(\omega)\}$$

In words: The $\dot{K}$ complexity of a sequence is the length of the shortest program able to learn to predict the sequence.

It can easily be seen that $\dot{K}$ has the same invariance to the choice of reference universal Turing machine as Kolmogorov complexity.

We also generalise this definition to sets of sequences.

Previous results written in terms of $\dot{K}$ complexity

We can now rewrite our previous results more succinctly:

$$\forall \omega \ : \ 0 \leq \dot{K}(\omega) \stackrel{+}{<} K(\omega),$$

and for sets of sequences of bounded complexity,

$$\forall n \in \mathbb{N} \ : \ n \stackrel{+}{<} \dot{K}(\mathcal{C}_n) \stackrel{+}{<} n + O(\log_2 n).$$

In words: The simplest predictor capable of predicting all sequences up to a Kolmogorov complexity of n, has itself a Kolmogorov complexity of roughly n.

Do some individual sequences demand complex predictors?

Or more formally, does there exist ω such that $\dot{K}(\omega) \approx K(\omega)$.

Theorem

$\forall n \in \mathbb{N}, \exists \omega \in \mathcal{C} : n \overset{+}{<} \dot{K}(\omega) \overset{+}{<} K(\omega) \overset{+}{<} n + O(\log_2 n)$.

In words: For all degrees of complexity, there exist sequences where the simplest predictor able to learn to predict the sequence is about as complex as the sequence itself.

Proof sketch: Essentially we create a meta-predictor that simulates *all* predictors of complexity less than n. We then show that there exists a sequence ω which the meta-predictor cannot predict and therefore neither can *any* predictor of complexity less than n, that is, $n \overset{+}{>} \dot{K}(\omega)$. The remainder of the proof mostly follows from earlier results.

What properties do high $\dot{K}$ sequences have?

If program q generates ω, let $t_q(n)$ be the number of computation steps performed by q before the n^{th} symbol of ω is output.

Lemma

$\forall \omega \in \mathcal{C}$, *if* $\exists q : \mathcal{U}(q) = \omega$ *and* $\exists r \in \mathbb{N}, \forall n > r : t_q(n) < 2^n$, *then* $\dot{K}(\omega) \stackrel{+}{=} 0$.

In words: If a sequence can be computed in a reasonable amount of time, then the sequence must have a low $\dot{K}$ complexity.

Proof sketch: Construct a predictor that on input $x_{1:n}$ simulates *all* programs of length n or less for 2^{n+1} steps each, then predicts according to the lexicographically first program with output consistent with $x_{1:n}$. So long as $t_q(n) < 2^n$ for the true generator, in the limit this predictor must converge to the right model.

No elegant powerful constructive theory of prediction exists

A constructive theory of prediction $\mathcal{T}$, expressed in some sufficiently rich formal system $\mathcal{F}$, is in effect a description of a prediction program with respect to a universal Turing machine which implements the required parts of $\mathcal{F}$.

Thus we can re-express our previous results,

Corollary

For individual sequences with high $\dot{K}$ complexity, and for the sets of all sequences of bounded Kolmogorov complexity, the predictive power of a constructive theory of prediction $\mathcal{T}$ is limited by $K(\mathcal{T})$.

This is in marked contrast to Solomonoff's highly elegant but *non-constructive* universal theory of prediction.

Gödel incompleteness and prediction

Theorem

In any formal axiomatic system $\mathcal{F}$ that is sufficiently rich to express statements of the form "$p \in P_n$", there exists $m \in \mathbb{N}$ such that for all $n > m$ and for all predictors $p \in P_n$ the true statement "$p \in P_n$" cannot be proven in $\mathcal{F}$.

In words: Even though we have proven that very powerful sequence prediction programs exist ($\forall n \in \mathbb{N} : P_n \neq \varnothing$), beyond a certain complexity it is impossible to find any of these programs using mathematics.

The proof has a similar structure to Chaitin's information theoretic proof of Gödel incompleteness.

Prediction incompleteness proof

Proof sketch: Create a search program s that searches for a proof of the statement "$p \in P_n$" and halts with output p if it finds such a proof.

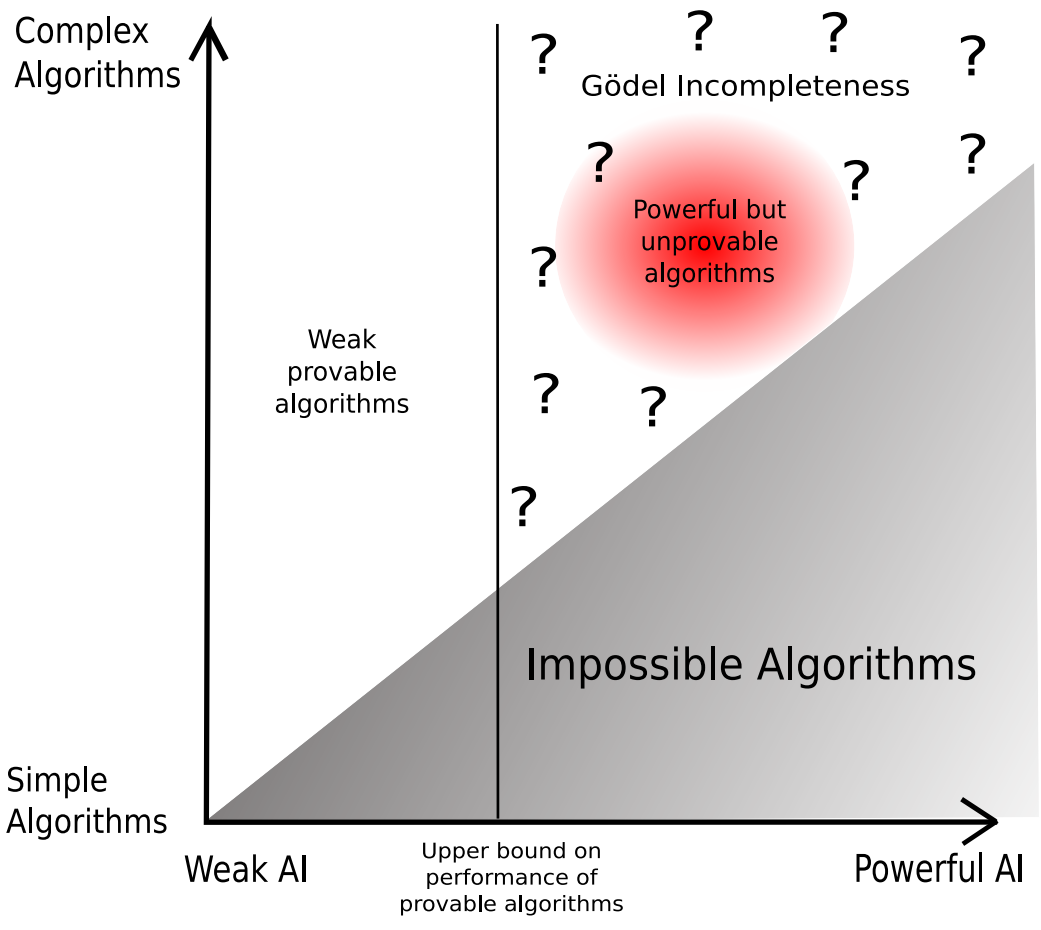
As s contains an encoding of n, and some constant length code we see that $K(s) \stackrel{+}{<} O(\log_2 n)$.

Assume that s finds a proof and halts with output p. This means that s is an effective description of p and thus, $K(p) \stackrel{+}{<} O(\log_2 n)$.

However we have shown earlier that $p \in P_n \Rightarrow K(p) \stackrel{+}{>} n$. Thus for large n we have a contradiction!

Therefore, for large n the true statement $p \in P_n$ cannot be proven.

Summary


Complex
Algorithms
Gödel Incompleteness
Powerful but
unprovable
algorithms
Weak
provable
algorithms
Impossible Algorithms
Simple
Algorithms
Weak AI
Upper bound on
performance of
provable algorithms
Powerful AI