Error in Enumerable Sequence Prediction

Extended abstract

Nick Hay

Department of Computer Science University of Auckland, New Zealand Email: nhat005@ec.auckland.ac.nz

Abstract. We outline a method for quantifying the error of a sequence prediction. With sequence predictions represented by semimeasures $\nu(x)$ we define their error to be $-\log_2\nu(x)$. We note that enumerable semimeasures are those which model the sequence as the output of a computable system given unknown input. Using this we define the simulation complexity of a computable system C relative to another U giving an exact bound on their difference in error. This error in turn gives an exact upper bound on the number of predictions ν gets incorrect.

1 A prediction's error

Suppose we wish to predict a sequence over a finite alphabet X.

Definition 1. [3] A semimeasure ν is a function $\nu: X^* \to [0,1]$ satisfying:

1. Normalisation:

$$\nu(\epsilon) = 1$$

2. Coherence:

$$\sum_{c \in X} \nu(xc) \le \nu(x)$$

 $\nu(x)$ is the probability the true sequence begins with x, for finite strings $x \in X^*$. The two conditions are both necessary and sufficient for this interpretation to make sense. In this way semimeasures are predictions of what the true sequence is.

The smaller the probability $\nu(x)$ that ν assigns to the sequence so far x, the more the prediction is in error. We thus define the error as a decreasing function of the probability $\nu(x)$:

Definition 2. The error $E\nu: X^* \to [0,\infty]$ of a prediction ν is

$$E\nu(x) = -\log_2 \nu(x)$$

Dagstuhl Seminar Proceedings 06051 Kolmogorov Complexity and Applications http://drops.dagstuhl.de/opus/volltexte/2006/633 The logarithm, as opposed to another decreasing function, is necessary for the following decomposition:

$$\begin{aligned} \mathbf{E}\nu(x) &= -\log \nu(x) \\ &= -\log \prod_{i=1}^{|x|} \nu(x_i|x_{< i}) \\ &= \sum_{i=1}^{|x|} -\log \nu(x_i|x_{< i}) \\ &= \sum_{i=1}^{|x|} \mathbf{E}\nu(x_i|x_{< i}) \end{aligned}$$

where the second step follows the product rule of probability theory. Note that:

- 1. x_i is the *i*th symbol of x; $x_{\leq i}$ is the first i-1 symbols of x.
- 2. $\nu(x_i|x_{< i}) = \nu(x_{< i}x_i)/\nu(x_{< i})$ is the probability the *i*th symbol is x_i , given that the predictor knows all the earlier symbols $x_{< i}$.

This reduces predictions of the whole sequence x to sub-predictions of each element of the sequence x_i . The total error $E\nu(x)$ is separated into errors $E\nu(x_i|x_{< i})$ for each of these sub-predictions. $E\nu(x)$ measures the cumulative error in the sub-predictions.

The decomposition gives intuition into what the error means. If a particular sub-prediction assigns more than probability 1/2 to the correct answer we consider the sub-prediction correct. This means all incorrect sub-predictions assign less than probability 1/2 and thus have an error greater than 1. As a direct result:

Lemma 1. A prediction ν has at most $|E\nu(x)|$ incorrect sub-predictions.

2 Predicting sequences generated by computable systems

Definition 3. [3] An enumerable semimeasure $\nu(x)$ is a semimeasure where the set

$$\{(x,p): p \le \nu(x), \ p \in [0,1] \cap \mathbb{Q}\}$$

is computably enumerable.

This means there is a computable process we can ask

"does ν predict x with at least probability p?"

and always receive an answer when it's "yes". Although all practially implementable semimeasures will be enumerable, the converse doesn't hold. We use

this large a class because it has a nice represention: enumerable semimeasures correspond to predictions about the output of a computable system.

A system C is a black box which can input bits $p_i \in 2$ and output symbols $x_j \in X$. The outputs and inputs may be interleaved in any way. Given a series of input bits, a system's output forms a sequence. Define a system C's behaviour

$$C: 2^* \to X^\#$$

where 2^* is the set of finite binary sequences and $X^{\#}$ the set of finite and infinite sequences over X. C(p) is the sequence of outputs C makes when p is the series of input bits available. We stop recording outputs if C tries to read more than p bits.

Definition 4. A computable system is a system with computable behaviour. In particular, its behaviour C can be implemented by a monotone Turing machine.

Equivalently, the behaviour $C: 2^* \to X^\#$ is computable when the following sets are computably enumerable:

1. $\{(p,x) \in 2^* \times X^* : x \le C(p)\}$

2.

$$\{(p,x)\in 2^*\times X^*: p \text{ is non-terminal for } C \text{ and } x=C(p)\}$$

A string $x \in 2^*$ is non-terminal for C if there exists $y \neq x$ such that C(x) < C(y). This means that C eventually outputs more symbols after receiving more input bits.

Functions satisfying the above axioms are called processes in [1].

Suppose we know nothing about how the input to the computable system C is generated: we assign equal probability to 0's and 1's. The probability the output begins with x is $\mu_C(x)$:

Definition 5. [2] A computable system C's **output semimeasure** is:

$$\mu_C(x) = \sum_{p:C(p)=x*} 2^{-|p|}$$

where C(p) = x* means p is a minimal input string such that C(p) begins with x (i.e. no prefix of p has output beginning with x).

Lemma 2. For any computable system C, the output semimeasure μ_C is enumerable.

Theorem 1. For any enumerable semimeasure $\nu(x)$ there exists a computable system C such that

$$\nu(x) = \mu_C(x)$$

Prediction using an enumerable semimeasure is equivalent to modelling the sequence as generated by a computable system fed an unknown input. See [3] for a similar result.

3 Complexity of prediction bounds error difference

Definition 6. A computable system U simulates a computable system C with constant $k_C \in \mathbb{N}$ if for all x there exists an injective f_x :

$$f_x: \{p: C(p) = x*\} \to \{q: U(q) = x*\}$$

such that

$$|f_x(p)| \le |p| + k_C$$

U simulates C if we can recode all the minimal input strings for C outputting x as minimal input strings for U doing the same without increasing their length too much.

Definition 7. The simulation complexity $S_U(C)$ of a computable system C relative to another U is

$$S_U(C) = \min\{k_C : U \text{ simulates } C \text{ with constant } k_C\}$$

This simulation complexity is the least amount of overhead the best simulation needs. We use this measure rather than a standard one to derive exact results.

Definition 8. The simulation complexity $S_U(\nu)$ of a semimeasure ν relative to U is:

$$S_U(\nu) = \min\{S_U(C) : \forall x \ \mu_C(x) = \nu(x)\}\$$

 $S_U(\nu)$ is the simulation complexity of the simplest computable system C which simulates ν (i.e. has ν as its output semimeasure).

Theorem 2. For computable systems U and C, and for all x:

$$E\mu_U(x) \le E\mu_C(x) + S_U(C)$$

Corollary 1. Given a computable system U, and any enumerable semimeasure ν :

$$E\mu_U(x) \le E\nu(x) + S_U(\nu)$$

References

- $1. \ \, {\rm Calude,\,2002.\,Information\,\,and\,\,Randomness.\,\,An\,\,Algorithmic\,\,Perspective,\,2nd\,\,Edition.\,\,Springer-Verlag.}$
- 2. Hutter, 2004. Universal Artificial Intelligence. Springer, Berlin.
- 3. Li and Vitanyi, 1997. An Introduction to Kolmogorov Complexity and its Applications, 2nd Edition. Springer-Verlag, New York.