

Error in Computable Sequence Prediction

Nick Hay

4th May 2006

Motivation:

- Predicting sequences (e.g. the inputs to an artificial intelligence) is important for manipulating the universe.
- Predictions with less error are more useful.
- Predictions must be computable to be practically useful.
- This motivates studying error in computable sequence prediction.

[Make this overview informal. The formal one is at the end. ?]

- We define the error $E\nu(x\downarrow)$ of a predictor ν as the negative log probability it assigns to the actual outcome x .
- We study predictors μ_C which assume the sequence is generated by a computer running an unknown program, without loss of generality.

We find:

- If the true output is x , μ_C has at most $K_C(x)$ error and makes at most that many mistakes.
- μ_C infers the environment's program "faster" than we can tell it.
- μ_C 's error is at most $K_C(\nu)$ greater than any other predictor ν 's, irrespective of the true output x .

We first present definitions, then move to results and analysis.

Semimeasures and computers:

- Semimeasures
- Enumerable semimeasures
- Computable systems (computers)
- Computer's output semimeasure
- Computers represent enumerable semimeasures
- Universal computers
- Complexity measures

- A **semimeasure** $\nu: X^* \rightarrow [0, 1]$ satisfies:

- Normalisation:

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

- Coherence:

$$\sum_{c \in X} \nu(\mathbf{x}c) \leq \nu(\mathbf{x})$$

- $\nu(\mathbf{x})$ is the probability the true sequence begins with \mathbf{x} , for finite sequences \mathbf{x} .
- ν uniquely determines the probability the true sequence is *exactly* \mathbf{x} , denoted $\nu(\mathbf{x}\downarrow)$:

$$\nu(\mathbf{x}\downarrow) = \begin{cases} \nu(\mathbf{x}) - \sum_{c \in X} \nu(\mathbf{x}c) & \text{for } \mathbf{x} \text{ finite} \\ \lim_{n \rightarrow \infty} \nu(\mathbf{x}_{1:n}) & \text{for } \mathbf{x} \text{ infinite} \end{cases}$$

- ν captures a **prediction** about what the actual sequence is.

Enumerable semimeasures

- An **enumerable semimeasure** $\nu(x)$ is a semimeasure where the set

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

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”.

- All practically implementable semimeasures will be enumerable, although the converse doesn’t hold.
- We use this large a class because it has a nice representation: *enumerable semimeasures correspond to predictions about the output of a computable system.*

Computable systems

- A **system** C can input bits $p_i \in 2$ and output symbols $x_j \in X$. They may interleave outputs and inputs in any way.
- Given input bits, a system's output forms a sequence.
- Define its **behaviour** $C: 2^* \rightarrow X^\#$. $C(p)$ is the sequence of outputs C makes when p is the sequence of input bits available. We stop recording outputs if C tries to read more than p bits.
- 2^* is the set of finite binary sequences, $X^\#$ the set of finite and infinite sequences over X .
- A **computable system** (computer) is a system with computable behaviour. This means its behaviour C can be implemented by a monotone Turing machine.
(Equivalently, its behaviour is a process.)

Computer's output semimeasure

- Suppose we know nothing about how the input to the computer C is generated: we assign equal probability to 0's and 1's.
- The probability the output begins with x is:

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

where $C(p) = x^*$ means p is a minimal sequence such that $C(p)$ begins with x .

- This sum is the probability that the unknown input sequence p has $C(p)$ beginning with x .

Lemma

For all computers C , the output semimeasure μ_C is enumerable.

Computers represent enumerable semimeasures

Theorem

For any enumerable semimeasure $\nu(x)$ there exists a computer C such that

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

Proof outline

The computer enumerates ν , allocating program outputs so the output semimeasure is at most 2^{-i} smaller than the enumeration of ν at step i .

- Prediction using an enumerable semimeasure is equivalent to modelling the environment as a computer with an unknown program.

Definition

A computer U is **universal** if for all computers C there exists $k_C \in \mathbb{N}$ such that for all p there exists q where:

1. $U(q) = C(p)$
2. $|q| \leq |p| + k_C$

We say U simulates C with constant k_C .

Definition

The complexity $K_U(C)$ of a computer C relative to another U is

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

- We introduce new complexity measures to avoid additive constants.

Definition

The complexity $K_U(x)$ of a sequence x relative to U is:

$$K_U(x) = \min\{|p| : \forall p' \geq p \ U(p') = x\}$$

- If we define computers which output x without reading any program bits, i.e., C_x where $\forall p \ C_x(p) = x$, we have $K_U(C_x) = K_U(x)$.

Definition

The complexity $K_U(\nu)$ of a semimeasure ν relative to U is:

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

- $K_U(\nu)$ is the complexity of the simplest computer C which simulates ν (i.e. has ν as its output semimeasure).

Error in enumerable prediction:

- The error of a prediction
- Conditional probability
- Partial errors
- Prediction mistakes
- Error bounds number of prediction mistakes
- Complexity of output bounds total error
- “Easy” to predict environments
- Universal prediction

The error of a prediction

- The semimeasure ν predicts the sequence will be exactly x with probability $\nu(x\downarrow)$. The lower the probability the less it “expects” the sequence to be x .
- If x is the actual sequence, the smaller this probability the more error in the prediction.

Definition

The **error** $E\nu(x\downarrow)$ of a prediction ν is

$$E\nu(x\downarrow) = -\log \nu(x\downarrow)$$

where x is the actual sequence.

- The logarithm is necessary for the partial error decomposition we introduce later.

Conditional probability

- $\nu(\mathbf{x}\downarrow)$, the probability the sequence is exactly \mathbf{x} , can be decomposed via probability theory (the product rule):

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

- x_i is the i th symbol of \mathbf{x} . $x_{<i}$ is the first $i - 1$ symbols of \mathbf{x} .
- $\nu(\downarrow|\mathbf{x})$ is the probability that the output is *exactly* \mathbf{x} , given we know the output *starts* with \mathbf{x} .
- $\nu(x_i|x_{<i})$ is the probability the i th symbol is x_i , given we know all the earlier symbols $x_{<i}$.
- This means the prediction that the entire output is \mathbf{x} can be broken into sub-predictions about individual outputs.

- The error can be similarly decomposed

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

- The total error of the prediction is spread over $|\mathbf{x}| + 1$ sub-predictions: predicting each element of the sequence and its termination.
- Each part of the sequence is predicted in order, with the sub-predictions informed by the earlier parts of the sequence.

Prediction mistakes

- It's natural to divide the $|x| + 1$ sub-predictions into those that are “mistakes”, and those that aren't. We want predictions with minimal mistakes.
- Mistaken predictions cannot assign a probability greater than $1/2$ to the correct answer.
- This is because a possibility assigned probability greater than $1/2$ is more likely than all other alternatives put together.
- So mistaken predictions have an error greater than or equal to $-\log 1/2 = 1$.

Error bounds number of prediction mistakes

Theorem

The predictor μ_C makes at most $\lfloor E\mu_C(\mathbf{x}\downarrow) \rfloor$ scattered prediction mistakes.

Proof

Since:

$$E\mu_C(\mathbf{x}\downarrow) = \sum_{i=1}^{|\mathbf{x}|} E\mu_C(\mathbf{x}_i | \mathbf{x}_{<i}) + E\mu_C(\downarrow | \mathbf{x})$$

at most $\lfloor E\mu_C(\mathbf{x}\downarrow) \rfloor$ sub-predictions can be in error, otherwise the RHS would be larger than the LHS.

- This is a weak bound! If the errors are large, e.g. a partial error of $\log |X|$ or more, there are much fewer mistakes.

Complexity of output bounds total error

Theorem

The total error $E\mu_C(x\downarrow)$ is bounded by the complexity of the output $K_C(x)$:

$$E\mu_C(x\downarrow) \leq K_C(x)$$

- **Proof outline:** The shortest program outputting x is one of the programs outputting x :

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

Corollary

μ_C makes at most $K_C(x)$ scattered errors in prediction.

- As $K_C(x)$ is the length of C 's shortest program generating x , everytime μ_C makes a mistake it learns at least one bit of C 's program!

“Easy” to predict environments

- We modify C to see if we can reduce the error by making it “easy to predict”.
- Take a computer C and modify it so that it tells us its program p before executing it:

$$D(p) = f(p)C(p)$$

for some injective encoding $f: 2^* \rightarrow X^*$.

- D tells you exactly what you need to know, its program, to exactly predict its future behaviour.

Theorem

For any program p the computers could be running, μ_C never has a larger error than μ_D :

$$E\mu_C(C(p)\downarrow) \leq E\mu_D(D(p)\downarrow)$$

- The predictor is *more* accurate if you don't tell it the computer's program!
- This is because μ_C doesn't need to infer the exact program p that C is running, only the equivalence class $[p]$ of programs with identical outputs.
- However, we know μ_D 's errors are concentrated in the first $|f(p)|$ bits, after which prediction is perfect, whereas μ_C 's errors are arbitrarily scattered.

Theorem

For U universal, any computer C , and for all x :

$$E\mu_U(x\downarrow) \leq E\mu_C(x\downarrow) + K_U(C)$$

- **Proof idea:** If $C(p) = x$ then there exists q where $U(q) = x$ and

$$|q| \leq |p| + K_U(C)$$

- Using a predictor μ_C with $K_U(C)$ extra bits of information about the environment results in a error at most $K_U(C)$ higher!
- μ_U 's "error pool" (to be distributed over its sub-predictions) is at most $K_U(C)$ bits larger than μ_C 's.

Corollary

For U universal and any enumerable semimeasure $\nu(x)$:

$$E\mu_U(x\downarrow) \leq E\nu(x\downarrow) + K_U(\nu)$$

- μ_U performs at most $K_U(\nu)$ bits worse than *any* other predictor ν .

- We modelled the sequence as generated by a universal computer U with an unknown program.
- We measured the accuracy of μ_U 's prediction of the computer's output sequence x by its error:

$$E\mu_U(x\downarrow) = -\log \mu_U(x\downarrow)$$

- This error measures the total error made in the prediction. It is spread over sub-predictions of individual elements of x .
- The error bounds the maximum number of sub-predictions that can be in error, although it doesn't restrict where the errors occur.

Finally, we described three results:

- If the true output is x the universal predictor μ_U 's error is at most $K_U(x)$, and makes at most that many mistakes:

$$E\mu_U(x\downarrow) \leq K_U(x)$$

- μ_U has a *lower* error if it is told the exact program U is running.
- The universal predictor μ_U 's error is no more than $K_U(\nu)$ larger than any other predictor ν 's:

$$E\mu_U(x\downarrow) \leq E\nu(x\downarrow) + K_U(\nu)$$

(The first expression is a special case of this one.)