Inductive Inference of Integer Sequences

Sam Tetruashvili
Advisor: Manuel Blum

April 12, 2010

Abstract

The goal of this senior thesis is design algorithms that can (inductively) infer integer sequences with high confidence, under the assumption that all terms of the sequence are accessible. We aim to bound the number sequence terms that an inference algorithm needs to see before it can make an inference it is confident in. We also aim to show that for certain sequences, our algorithm can give strong evidence that it cannot infer the sequence.

1 Introduction

The term "inductive inference" denotes the process of hypothesizing a general rule from examples. [AS83] In this work, we study the problem of inductive inference in the context of integer sequences. To facilitate this endeavour we define a natural, and hopefully very general, model for studying this problem. With this model in place we move on to try to characterize the set of sequences that can be efficiently inferred. We then move on to evaluate the performance of the model by trying to infer the sequences catalogued in the Sloane On-Line Encyclopedia of Integer Sequences [Slo].

It should be noted that much of this work is very different from modern statistical machine learning. One major difference is that we assume absolutely no noise in our data. Given that our data is error free, we take up the task of producing error-free inferences. This means that the hypotheses our model generates should be entirely consistent with all of the training data that the model has seen. Furthermore, a hypothesis that differs from the entire infinite integer sequence in exactly one instance is *just as bad* as a that hypothesis gets every term of the integer sequence wrong.

2 The Evolutionary Inference Model

We now present a general model for inferring algorithmically generated integer sequences. This model has its roots in the scientific method. At a very high level the model makes and tests hypotheses. When it finds that its hypothesis disagrees with the input sequence, it simply makes a new hypothesis using all of the data it has seen so far. It is important to note that these hypotheses must be *fully consistent* with the sequence terms that have been seen so far.

Definition 1. We say that an integer sequence, $\{a_t\}_{t\geq 0}$, is *algorithmically generated* if there is a Turing Machine, M, that on input $t \in \mathbb{N}$ halts with output a_t for all $t \geq 0$. For the remainder of this paper we will use the terms sequence and algorithmically generated integer sequence interchangeably.

Definition 2. The *Evolutionary Inference Model* for inferring algorithmically generated integer sequences has three main components:

1. The *algorithmically generated* integer sequence, $\{a_t\}_{t\geq 0}$ that is to be inferred. We require that this sequence be represented in a way that allows our algorithm to efficiently retrieve a_t at time t.

2. The *inference algorithm*, A, that we are using to infer $\{a_t\}_{t\geq 0}$. This inference algorithm generated hypotheses, $\{h_t\}_{t\geq 0}$, that come from some well defined *concept class*. For example, we present an inference algorithm that tries to infer the input sequence as a linear recurrence. It should be noted that any hypothesis output by an inference algorithm should be fully consistent with all the (finite amount of) data that has been input.

3. The *confidence function*, C, that tells us how confident we are in the current hypothesis, $\{h_t\}_{t\geq 0}$, given that we are currently at time t of our algorithm. The range of these confidence functions is the interval $[0, 1]$, where 0 indicates no confidence and 1 indicates total confidence. It should be noted that C is allowed to depend on the concept class that our inference algorithm is based on.

```
1 :  EvolutionaryInference(\{a_t\}_{t≥0}, A, C, ε)
2 :    \{h_t\}_{t≥0} = \{0\}_{t≥0}
3 :
4 :    for t = 0, 1, 2, ... do
5 :      if h_t ≠ a_t then
6 :        \{h_t\}_{t≥0} = A([a_0, a_1, ..., a_t])
7 :      else if C(\{h_t\}_{t≥0}, t) ≥ 1 − ε then
8 :        break
9 :
10 :   return \{h_t\}_{t≥0}
```

Figure 1: Pseudo-code for the Evolutionary Inference Model.

Given these components, the model will then proceed to try to use the inference algorithm to find a hypothesis that it is $1-\epsilon$ confident in, for some input $\epsilon > 0$, as shown in Figure 1.

Given some $\epsilon > 0$, the goal of this model is to be able to find a hypothesis that the model is $1-\epsilon$ confident in. In the remainder of this work we strive to design inference algorithms and confidence functions that allow us to make sure that the model does not become too confident in an incorrect hypothesis.

3 Inference Algorithms

3.1 Polynomials

The simplest concept class of sequences that we study are those generated by a polynomial with rational coefficients.

Definition 3. We say that a sequence, $\{a_t\}_{t\geq 0}$, *is generated by a polynomial of degree k* if there are rational numbers $c_0, c_1, \ldots, c_k$ such that

$$a_t = \sum_{i=0}^{k} c_i t^i$$

for all $t \geq 0$.

The inference task here is to find the degree and coefficients (c_i's) of the polynomial that generated our sequence.

Theorem 1. Let $\{a_t\}_{t\geq 0}$ be generated by a polynomial of degree k. We can infer $\{a_t\}_{t\geq 0}$ given at least its first $k+1$ terms. Furthermore this bound is tight.

Proof. Given in thesis along with an algorithm that achieves the bound.

3.2 Linear Recurrences

The next simplest class of sequences we study are those generated by a linear recurrence with rational coefficients.

Definition 4. We say that a sequence, $\{a_t\}_{t\geq 0}$, *is generated by a linear recurrence of degree k* if there there are integers $b_0, b_1, \ldots, b_{k-1}$ and rationals $c_0, c_1, \ldots, c_{k-1}$ such that for all $t < k$ $a_t = b_t$ and for all $t \geq 0$

$$a_{t+k} = \sum_{i=0}^{k-1} c_i a_{t+i}$$

Much like with polynomials, the inference task here is to find the degree, base cases (b_i's), and coefficients (c_i's) of the linear recurrence that generated our sequence.

Theorem 2. Let $\{a_t\}_{t\geq 0}$ be generated by a linear recurrence of degree k. We can infer $\{a_t\}_{t\geq 0}$ given at least its first $2k$ terms.

Proof. Given in thesis along with the algorithm that achieves the bound.

3.3 Decimal Expansions

The class of decimal expansions is very large. I've broken this class up into three slightly simpler classes. The first class is the set of all decimal expansions of rational numbers. [BBS82] gives an algorithm for efficiently inferring sequences in this class. The second class is the set of algebraic numbers (numbers that can be represented as the root of a polynomial with integer coefficients). Finally the third class is the set of numbers that can be expressed as some linear combination of some real numbers with known decimal expansions. For example, say we know the decimal expansion of π, ϕ, e, γ, and $\sqrt{2}$. We can then infer the decimal expansion of numbers like

$$a = 2\pi + 10e - 34\phi + \sqrt{2}$$

The inference algorithm for the last two classes use the LLL lattice reduction algorithm [LLL82] to find the necessary coefficients.

For the sake of brevity we will defer a deeper treatment of these 3 classes of sequences to the final thesis document as the analysis gets fairly involved. It suffices to say we have inference algorithms for each of these three classes along with an upper bound on the number of terms our inference algorithms need to see before they can make a correct inference.

3.4 Automatic Sequences

We now consider inferring automatic sequences. This class is fairly general. At a high level you can think of these sequences as those that can be generated by a Deterministic Finite Automaton (DFA). Say you give the DFA the binary representation of a non-negative integer t and the DFA output 0 or 1 for a_t.

Definition 5. A Partitioned Deterministic Finite Automaton (PDFA) is an automaton of the form

$$M = (Q, \Sigma, \delta, q_0, \mathbf{F})$$

where (Q, Σ, δ) is a deterministic and complete transition system, $q_0 \in Q$ is an initial state, and $\mathbf{F} = (F_a \mid a \in \Delta)$ is a partition of Q. We take Δ to be some output alphabet.

Example. This notion generalizes DFAs. We can represent every DFA

$$M = (Q, \Sigma, \delta, q_0, F)$$

as the PDFA

$$M' = (Q, \Sigma, \delta, q_0, (Q - F, F))$$

With $\Delta = \{0, 1\}$. The partition $(Q - F, F)$ simply says that if the final state is in the set $Q - F$ the PDFA output a 0, otherwise if the final state is in F the PDFA output a 1.

Definition 6. We say that a sequence, $\{a_t\}_{t\geq 0}$, *is generated by a PDFA of degree* (k, m) if there is a PDFA with m states and input alphabet $\{0, 1, \ldots, k-1\}$ such that for all $t \geq 0$ the PDFA output a_t when run on the k-ary representation of t.

The inference task here is to find the smallest PDFA that generates our sequence.

Theorem 3. Let $\{a_t\}_{t\geq 0}$ be generated by a PDFA of degree (k, m). We can infer $\{a_t\}_{t\geq 0}$ given at least the first k^{2m-2} terms of the sequence. Furthermore this bound is tight.

Proof. Given in thesis along with an algorithm that achieves the bound.

3.5 Turing Machines

We now consider the most general class of sequences: those that can be generated by a program in any modern programming language.

Definition 7. We say that a sequence, $\{a_t\}_{t\geq 0}$, *is generated by a Turing Machine of degree* k, if there is a k state Turing Machine that on input t outputs a_t.

We now give an impossibility result that suggests that this class of sequences cannot be inferred in general.

Definition 8. The Busy-Beaver function, $\beta : \mathbb{N} \to \mathbb{N}$, is defined such that for every positive integer n $\beta(n)$ is equal to maximum number of 1's a Turing Machine on n states can print given that it halts.

Fact 4. For any computable function $f : \mathbb{N} \to \mathbb{N}$, there is $x \in \mathbb{N}$ such that $f(x) < \beta(x)$.

Definition 9. Let $f : \mathbb{N} \to \mathbb{N}$ be the function that for each positive integer k tells us the minimum number of sequence terms we need to see before we can infer a sequence generated by a Turing Machine of degree k.

Theorem 5. $\forall k \in \mathbb{N}\ f(k) > \beta(k)$.

Proof. Given in thesis.

Corollary 6. f is not computable.

Proof. By Fact 4.

This result suggests that any inference algorithm for Turing Machine sequences needs to be given a lot of sequence terms before it can make an inference.

4 Confidence Functions

The role of confidence functions in our model is to let us know when we are confident enough in a hypothesis that we can stop the inference procedure and assert that the hypothesis generates the input sequence. This is arguably the most theoretically interesting, and difficult, part of this work. What we've strived to do is try to define confidence functions by a certain set of properties that we would like them to have. In this process we've come up with the following two properties.

Property 1 (Monotonicity)**.** Say we make some hypothesis, $\{h_t\}_{t\geq 0}$, at some time t and we've kept this hypothesis until some time $t' > t$. Then any confidence function, C, should have

$$C(\{h_t\}_{t\geq 0}, t') \geq C(\{h_t\}_{t\geq 0}, t)$$

In other words, our confidence in a hypothesis cannot decrease until we've found a counterexample to the hypothesis.

Property 2 (Convergence)**.** Say we make some hypothesis, $\{h_t\}_{t\geq 0}$, at some time t_0 and this hypothesis actually does generate the input sequence. Then our confidence in this hypothesis should tend to 1 as t tends to infinity. More formally

$$\lim_{t\to\infty} C(\{h_t\}_{t\geq 0}, t_0 + t) = 1$$

A simple confidence function with these properties is the fraction of terms that we've used to check our current hypothesis. For example, Theorem 1 says that we only need $k+1$ sequence terms to infer a sequence generated by a polynomial of degree k. Thus if we are at time t and our current hypothesis is a polynomial of degree k then we know that we have checked this hypothesis on exactly $t+1-(k+1) = t-k$ sequence terms. Then we can use the following confidence function

$$C(\{h_t\}_{t\geq 0}, t) = \frac{t-k}{t+1}$$

Where $\{h_t\}_{t\geq 0}$ is a polynomial hypothesis of degree k. This confidence function clearly has both of the desired properties.

We can make similar confidence functions for our other concept classes, but so far as we can tell the two properties we have are too weak. The direction we've decided to go is to look for properties of each of our concept classes that we can exploit to design good confidence functions. For example, one can study the probability that a random sequence of $2k$ integers is describable by a linear recurrence of degree less than k. It turns out that this probability tends to 0 as k tends towards infinity. This gives us the intuition that the marginal confidence gained by checking a hypothesis on one more sequence term must be a concave function (I call this property diminishing returns). Given concept classes with this property we can design our confidence functions for these concept classes to have an additional concavity property.

Unfortunately none of the confidence functions we've come up with so far seem to give any observable improvement over simply observing the number of terms we've used to check our current hypothesis. We do believe there is much more work that can be done in this area to achieve far better results than the simple confidence functions we use.

5 The On-Line Encyclopedia of Integer Sequences

We evaluate the quality of our model by using the data in Sloane's On-Line Encyclopedia of Integer Sequences. This is a truly wonderful resource that contains sequences that have come up in the course of academic research. You can think of this database as containing the set of "interesting sequences," meaning sequences which people have been interested in. For example, there is an entry for the Busy-Beaver sequence, the sequence of stops the A train makes in New York City, the Collatz sequence, etc.

As you may have already guessed most of the sequences in this database should be fairly difficult, if not impossible, to infer using the techniques we have outlined in this work. In fact the task of inferring these sequences is sometimes isomorphic to some of the most difficult open problem in modern mathematics: for example consider sequence A001359 which is the sequence whose t^{th} term is the smaller prime in the t^{th} twin prime pair. Successfully inferring this sequence would solve the Twin Prime Conjecture.

Thankfully this database also contains more tractable integer sequences that our methods are applicable to. It is our goal to be able to infer about 25% of the sequences in this database using the techniques in this work. So far we've been able to use our techniques to infer about 18.7% of the sequences in this database. A brief summary of our results is given in Figure 2.

We hope to modify some of our inference algorithms for decimal expansions to be able to infer a larger percentage of the database. This looks to be possible given that decimal expansions make up about 8% of the database. We can also improve this result by designing better confidence functions for each of our concept classes.

You can currently use our integer sequence inference system by going to `http://theory.res.cmu.edu/~samt/sequences.html`. At this website you can input the initial portion of a sequence and see if any of my inference algorithm can infer it. As a disclaimer this website is currently running off of my laptop so it may be off-line sometimes: namely when my laptop is not plugged via ethernet.

Concept Class	Number Inferred	Percent Inferred
Polynomial	4901	2.818 %
Linear Recurrence	29197	16.829 %
Decimal Expansions	2220	1.280 %
Automatic Sequences	1828	1.051 %
Total	32582	**18.734 %**

Figure 2: Inference statistics as of April 1, 2010 when the database had 173922 integer sequences.

References

[Ang74] Dana Angluin. Easily Inferred Sequences. Technical report, University of California at Berkeley, Department of EECS, 1974.

[AS83] Dana Angluin and Carl H. Smith. Inductive Inference: Theory and Methods. *ACM Computing Surveys*, 1983.

[AS03] Jean-Paul Allouche and Jeffrey Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, first edition, 2003.

[BBS82] Lenore Blum, Manuel Blum, and Michael Shub. Comparison of Two Pseudo-Random Number Generators. *Advances in Cryptology-Proceedings of Crypto '82*, 1982.

[Knu97] Donald E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Professional, third edition, 1997.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 1982.

[Slo] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. http://www.research.att.com/njas/sequences/.

[Wil94] Herbert S. Wilf. *generatingfunctionology*. Academic Press, second edition, 1994.