Solomonoff Induction

Shane Legg

May 16, 2004

Contents

Introduction

Solomonoff's induction method is an interesting theoretical model of what could be considered a perfect inductive inference system. Furthermore, it can be proven that a whole range of commonly used induction principles are computable approximations or special cases of Solomonoff's method. As such, Solomonoff induction provides us with a powerful and unifying perspective on the many diverse principles and methods that exist to deal with induction problems.

The foundations of Solomonoff induction are well rooted in the mathematics of computation and information theory. Indeed, due to his early work on induction, Solomonoff is now considered the father of the field of algorithmic information theory; a field which has exposed many deep connections between topics such as randomness, computability, complexity, chaos and Gödel incompleteness. It is perhaps surprising then that in many fields which deal with induction problems, for example statistics, Solomonoff's work on induction is almost completely unknown. It would seem that one of the reasons for this lies in the diverse range of background material demanded of the reader. For example, readers from a statistical background, while familiar with Bayes' theorem and measure theory, are usually unfamiliar with topics such as coding theory, computability theory and algorithmic information theory. Similarly readers from other disciplines are not usually equipped with all the necessary background.

As such, the first part of this report is spent briefly covering all the necessary background topics. In this way we hope to make the subject available to as wide an audience as possible. In the second part we present the basic essentials of Solomonoff's inductive inference method. The approach taken is not that originally used by Solomonoff but rather we come from the more general perspective of dominant enumerable semi-measures with Solomonoff's prior being a special case.

Because the purpose of this report is simply to serve as an introduction to Solomonoff's approach to inductive inference and not as an introduction to general inductive inference theory, it is reasonable to assume that the reader is already familiar with Bayes' theorem and the problems associated with selecting prior distributions. In fact one can view Solomonoff's inference method to be essentially just a general purpose Bayesian inference system with a special information theoretic universal prior.

Some effort has gone into keeping this work as small as possible without becoming terse. As such, many interesting connections to other topics and related issues are absent. This has been done to keep the demands on the reader (and myself!) to a minimum. Most of the proofs I have either extensively reworked myself or are of my own creation. This has further allowed me to reduce the complexity of the key results and should also add a more homogeneous feel to the material.

Acknowledgements

This work draws extensively and quite liberally from two main sources, namely Cristian Calude's book "Information and Randomness" [1] and Ming Li and Paul Vitányi's book "An Introduction to Kolmogorov Complexity and its Applications" [3]. The first book, while not dealing with inductive inference did however provide much of the necessary background material in a very rigorous and concise form. In particular the last two sections of the first chapter follow this book. All of the theorems in the first chapter appear in this book in some form.

The book by Li and Vitányi provided most of the material on Solomonoff induction itself. This book is fairly readable and covers an enormous range of topics related to algorithmic information theory (Kolmogorov complexity) and examines many connections to other areas of research. This book will not doubt become a classic in the field.

All of the theorems and lemmas in the second chapter appear either in this book or in their paper [2]. However, except for corollary 2.4.1 and theorems 2.5.1 and 2.6.1, the actual proofs in this text are either of my own creation or are significantly reworked versions of proofs from their book.

The main result of this work (theorem 2.5.1) was originally due to Ray Solomonoff. The original proofs of many other results which appear here, such as the invariance theorem, are also due to Solomonoff and appear in some form in either [4] or [5]. The proof of theorem 2.5.1 which we use is due to P. Gács and seems to only appear in [3].

For those interested in a very detailed look at dominant enumerable semimeasures form slightly different perspective to that given here, I recommend Zvonkin and Levin's paper [6].

1 Prerequisite Material

This chapter briefly covers the prerequisite material necessary for us to be able to study Solomonoff's inductive inference method. We start by clarifying our usage of mathematical notation.

1.1 Mathematical Notation

Let $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{R}$ represent the natural, rational, integer and real numbers respectively. Wherever possible the variable names i, j, k, l, m and n will be used for integers variables. The symbol $\equiv$ is used to indicate that two expressions are equal by definition. For example, define $\mathbb{R}^+ \equiv \{x \in \mathbb{R} : x \geq 0\}$. The minimum of an empty set is defined to be ∞. The symbols $\subset$ and $\subseteq$ express the strict subset and subset relations respectively. Define $A \setminus B \equiv \{a \in A : a \notin B\}$. Let $\wp(X) \equiv \{A : A \subseteq X\}$ be the *power set* of X and let $\#X$ be the cardinality of the set X. So for example, if $X = \{0, 1, 9\}$ then $\#X = 3$ and $\wp(X) = \{\emptyset, \{0\}, \{1\}, \{9\}, \{0, 1\}, \{0, 9\}, \{1, 9\}, \{0, 1, 9\}\}$. A countable subset of the power set of X , written $\{E_n\} \in X$, is called *pairwise disjoint* if for all $n \neq m$, $E_n \cap E_m = \emptyset$. If it is also the case that $E_1 \cup E_2 \cup \cdots = X$, then we call the collection of sets $\{E_n\}$ a *partition* of X. For example, the collection of sets of the form $(i, i + 1]$ where $i \in \mathbb{Z}$ form a partition of $\mathbb{R}$.

A *partial function* ψ, written $\psi : X \xrightarrow{o} Y$, is a function defined on a set $Z \subseteq X$, where Z is called the *domain* of ψ and is denoted $dom(\psi)$. If $X = dom(\psi)$ we say that ψ is a *total function*, or more simply a *function*, and we indicate this by writing $X \to Y$. For $x \notin dom(\psi)$ we put $\psi(x) = \infty$. The set $\{\psi(x) : x \in dom(\psi)\}$ is called the range of ψ and is denoted $range(\psi)$. Two partial functions $\psi, \phi : X \xrightarrow{o} Y$ are said to be *equal* iff $dom(\psi) = dom(\phi)$ and for all $x \in dom(\psi)$, $\psi(x) = \phi(x)$. For two functions $f : Y \to Z$ and $g : X \to Y$ define the composition of f and g to be $f \circ g(x) \equiv f(g(x))$.

1.2 Strings and Codes

An *alphabet* is a finite set with cardinality at least two. We call the elements of an alphabet *symbols*. For example the sets $\{a, b, c, d, \ldots, z\}$, $\{0, 1, 2, 3, \ldots, 9\}$ and $\{R, 7, x\}$ are all alphabets. Throughout this report we will make use of an arbitrary alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_Q\}$. Thus henceforth we will use Q to represent the number of symbols in our alphabet $\mathcal{A}$.

By $\mathcal{A}^n$ we mean the usual n-fold Cartesian product of sets,

$$\mathcal{A}^n \equiv \prod_{i=1}^{n} \mathcal{A}.$$

Thus $x \in \mathcal{A}^n$ is an ordered n-tuple: $x = (x_1, x_2, \ldots, x_n)$ where each $x_i \in \mathcal{A}$. Now define

$$\mathcal{A}^+ \equiv \bigcup_{n=1}^{\infty} \mathcal{A}^n$$

and call $\lambda \equiv \emptyset$ the *empty string*. Further define

$$\mathcal{A}^* \equiv \{\lambda\} \cup \mathcal{A}^+$$

and call $x \in \mathcal{A}^*$ a *string*. Wherever possible the variable names x, y and z will be used for arbitrary strings.

We denote the binary operation of *concatenation* over strings by juxtaposition and define the operation as the Cartesian product of the two strings; that is,

$$xy \equiv x \times y.$$

Thus if $x = (x_1, \ldots, x_n) \in \mathcal{A}^n$ and $y = (y_1, \ldots, y_m) \in \mathcal{A}^m$ then we define $xy = (x_1, \ldots, x_n, y_1, \ldots, y_m) \in \mathcal{A}^{n+m}$. It can easily be seen that if $x, y, z \in \mathcal{A}^*$ then,

$$xy \in \mathcal{A}^*,$$

$$x(yz) = (xy)z = xyz$$

and

$$\lambda x = x\lambda = x.$$

In other words, $\mathcal{A}^*$ is closed under concatenation, concatenation is associative and λ is the identity element. As concatenation is associative and not commutative we can adopt a more compact notation and simply write $x = x_1x_2x_3\cdots x_n$ for an arbitrary string.

If $x \in \mathcal{A}^n$ then we define $|x| \equiv n$ and call this the length of the string x. In particular $|\lambda| = 0$ and we see that

$$\forall x, y \in \mathcal{A}^* \quad |xy| = |x| + |y|.$$

As $x \in \mathcal{A}^n$ iff $|x| = n$, we will often use the notation $|x| = n$ to denote strings from $\mathcal{A}^n$ as it is more compact in large equations.

Every total ordering on $\mathcal{A}$, say $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_Q$, induces a quasi-lexicographical order on $\mathcal{A}^*$:

$$\lambda < a_1 < a_2 < \cdots < a_Q < a_1a_1 < a_1a_2 < \cdots$$

$$< a_1a_Q < a_2a_0 < \cdots < a_1a_1a_1 < \cdots < a_Qa_Qa_Q < \cdots .$$

Let $string : \mathbb{N} \to \mathcal{A}^*$ be the bijective function such that $string(n)$ is the nth string according to the above quasi-lexicographical order on $\mathcal{A}^*$.

Definition 1.2.1 *We say that a string $x \in \mathcal{A}^*$ is a* prefix *of a string $y \in \mathcal{A}^*$ iff*

$$\exists z \in \mathcal{A}^* \quad y = xz$$

and denote this $x \leq_p y$. We say that a set $S \subset \mathcal{A}^*$ is prefix free *iff*

$$\forall x, y \in S \quad x \leq_p y \Rightarrow x = y.$$

For example, if $\mathcal{A} = \{a, b, c, \ldots, z\}$ then *bed* $\leq_p$ *bedroom* and *aaza* $\leq_p$ *aazaaza*.

Definition 1.2.2 *Let $B = \{b_1, b_2, \ldots\}$ be a finite or infinite set. A* code *is an injective function $\psi : B \to \mathcal{A}^*$. We call the elements of $range(\psi)$* code-strings. *If the set of code strings is prefix-free we say that psi is an* instantaneous code.

Instantaneous codes are particularly important in practice as the prefix-free quality of the code strings allows a decoder to determine any particular code string without having to read beyond the end of the code string. Another useful property is the following elementary inequality:

Theorem 1.2.1 (Kraft Inequality) *If $n_1, n_2, \ldots \in \mathbb{N}$ are the lengths of code-strings of an instantaneous code $\phi : B \to \mathcal{A}^*$ then*

$$\sum_{i=1}^{\infty} Q^{-n_i} \leq 1.$$

Proof. To appear... □

By $\mathcal{A}^\omega$ we mean the countably infinite Cartesian product of $\mathcal{A}$; that is,

$$\mathcal{A}^\omega \equiv \prod_{i=1}^{\infty} \mathcal{A}.$$

Alternatively we could have defined $\mathcal{A}^\omega \equiv \{x_1 x_2 x_3 \cdots : x_i \in \mathcal{A}\}$. We call $\mathbf{x} \in \mathcal{A}^\omega$ a *sequence* and denote it in bold face. Intuitively a sequence is like a string of infinite length. Individual sequences will not be of much use to us in what follows; rather we will be interested in various sets of sequences. Of particular interest will be sets of all sequences which have a common string at their beginning, that is, sets of the form

$$x\mathcal{A}^\omega \equiv \{x_1 x_2 \ldots x_n y_1 y_2 y_3 \ldots : y_i \in \mathcal{A}\},$$

where $x = x_1 \ldots x_n \in \mathcal{A}^*$.

1.3 Measure Theory

While probability theory over discrete spaces is very well known, the mathematics for continuous spaces is fairly specialised. For this reason we give a very brief outline of the concepts we will need. There are countless books on this topic so the interested reader will have no trouble locating further information.

A probability function, or in this context a *probability measure*, is a function which assigns probabilities to various subsets of a sample space Ω. These subsets of Ω form what is called a *σ-algebra*.

Definition 1.3.1 *A collection $\mathcal{D} \subseteq \wp(\Omega)$ is called a* σ-algebra *on Ω iff*

$$\emptyset \in \mathcal{D},$$

$$A \in \mathcal{D} \Rightarrow \bar{A} \in \mathcal{D}$$

and

$$\{A_n\} \in \mathcal{D} \Rightarrow \bigcup_{n=1}^{\infty} A_n \in \mathcal{D}.$$

If $\mathcal{D}$ is a σ-algebra over Ω then we call the ordered pair $(\Omega, \mathcal{D})$ a *measurable space*. Clearly for any set Ω, both $\wp(\Omega)$ and $\{\emptyset, \Omega\}$ are σ-algebras. Less trivial σ-algebras are more difficult to explicitly define. Often when we want a σ-algebra on a space we already have some collection of subsets of the space and we would like the σ-algebra to include these sets. The following theorem is useful in this situation.

Theorem 1.3.1 *If $\mathcal{G} \subseteq \wp(\Omega)$ and $\mathcal{G} \neq \emptyset$ then there exists a unique σ-algebra $\sigma(\mathcal{G})$ which is the smallest σ-algebra such that $\mathcal{G} \subseteq \sigma(G)$.*

Proof Sketch. It can easily be shown that the intersection of any number of σ-algebras is also a σ-algebra. Thus we can simply define $\sigma(G)$ to be the intersection of all σ-algebras which are super sets of $\mathcal{G}$. □

Now that we have the basic concepts of σ-algebras and a method to construct them we now turn our attention to the functions which operate on these spaces.

Definition 1.3.2 *Let (Ω, D) be a measurable space. A function $\mu : \mathcal{D} \to \mathbb{R}^+$ is a* measure *iff for all mutually disjoint $\{E_n\} \in \mathcal{D}$ we have*

$$\mu\left(\bigcup_{n=1}^{\infty} E_n\right) = \sum_{n=1}^{\infty} \mu(E_n).$$

If has the additional property that (Ω) = 1 then we call a probability measure.

If is a measure over a measurable space $(\Omega, \mathcal{D})$, then we call the tuple (Ω, D) a *measure space.* If is also a probability measure on this measurable space, (Ω, D, μ) is called a *probability space.*

A similar, but somewhat less intuitive concept is that of a *semi-measure.* Semi-measures are not a part of classical measure theory but they are useful when considering certain computability aspects of measures.

Definition 1.3.3 *Let (Ω, D) be a measurable space. A function $\mu : \mathcal{D} \to \mathbb{R}^+$ is a* semi-measure *iff $\mu(\Omega) \leq 1$ and for all mutually disjoint $\{E_n\} \in \mathcal{D}$ we have*

$$\mu\left(\sum_{n=1}^{\infty} E_n\right) \geq \sum_{n=1}^{\infty} \mu(E_n).$$

Thus we can see that the class of probability measures is a subset of the class of semi-measures. One can think of a semi-measure which isn't a probability measure as being some kind of "defective" probability measure. Shortly we will examine a method for building semi-measures up to be probability measures.

1.4 Spaces of Sequences and Strings

Now that we have the basic rudiments of measure theory we now consider how this applies in our context of strings and sequences. We will soon be interested in predicting digits in a sequence after having seen a finite number of initial digits. This means that we need to have probability measures defined over sets of sequences which have common initial digits. Thus we require a σ-algebra which contains the following collection of sets;

$$\mathcal{P} \equiv \{x\mathcal{A}^\omega : x \in \mathcal{A}^*\} \cup \{\emptyset\}.$$

By theorem 1.3.1 we can simply define $\mathcal{S} \equiv \sigma(\mathcal{P})$ to get the required σ-algebra on $\mathcal{A}^\omega$.

While the creation of the measurable space $(\mathcal{A}^\omega, \mathcal{S})$ puts us on technically secure ground when considering the probability of various sets of sequences, it is still the case that for the purposes of induction we are really only interested in the probabilities of elements of $\mathcal{P}$. Thus it seems reasonable to save oneself the difficulties in working with $\mathcal{S}$ by restricting our analysis to the more simplistic space $\mathcal{P}$. Indeed, certain equivalence results exist that allow us to do this. It is clear from the definition of $\mathcal{P}$ that $\mathcal{P} \setminus \{\emptyset\}$ is isomorphic to $\mathcal{A}^*$. Furthermore, the following result holds;

Theorem 1.4.1 *There exists a bijective correspondence between the probability measures defined on $\mathcal{S}$ and the functions $h : \mathcal{A}^* \to [0,1]$ such that*

$$h(\lambda) = 1,$$

and

$$\forall x \in \mathcal{A}^* \quad h(x) = \sum_{|a|=1} h(xa).$$

Proof. To appear... □

A similar result holds for semi-measures. This means that we can investigate probabilities over $\mathcal{S}$ by looking at functions over $\mathcal{A}^*$. With these results in mind we can make the following definitions for functions over $\mathcal{A}^*$.

Definition 1.4.1 *A function $\mu : \mathcal{A}^* \to [0,1]$ is a* probability measure *iff,*

$$\mu(\lambda) = 1$$

and

$$\forall x \in \mathcal{A}^* \quad u(x) = \sum_{|a|=1} \mu(xa).$$

Definition 1.4.2 *A function $\mathcal{A}^* \to [0,1]$ is a* semi-measure *iff,*

$$\mu(\lambda) \leq 1$$

and

$$\forall x \in \mathcal{A}^* \quad \mu(x) \geq \sum_{|a|=1} \mu(xa).$$

Finally let us examine further the relationship between probability measures and semi-measures in this new context. We may create a probability measure μ from a semi-measure ρ by adding an extra symbol 'u' to the alphabet as follows. Firstly; as we want μ to be a probability measure, it must be the case that $\mu(\lambda) = 1$ and

$$\forall x \in \mathcal{A}^* \quad \mu(x) = \sum_{|a|=1} \mu(xa) + \mu(xu).$$

We also want μ and ρ to coincide over $\mathcal{A}^+$, thus set $\mu(x) \equiv \rho(x)$ for all $x \in \mathcal{A}^+$. It now follows that

$$\forall x \in \mathcal{A}^* \quad \mu(xu) = \rho(x) - \sum_{|a|=1} \rho(xa).$$

The problem is that a semi-measure can't tell us what is going on for strings that have u's which are not at the end of a string. There simply isn't enough information in the semi-measure. Hence the extension of the semi-measure to a probability measure by this method is non-unique. Nevertheless, this method will be sufficient for our purposes.

1.5 Kullback Divergence

Kullback divergence measures how much two measures differ from each other and so will be useful to analyse the difference in predictive accuracy between using a universal prior and the true prior.

Definition 1.5.1 *The* Kullback divergence *of a measure μ with respect to a semi-measure ρ is defined as*

$$D(\mu||\rho) \equiv \sum_{|a|=1} \mu(a) \ln \frac{\mu(a)}{\rho(a)}.$$

We will generalise this further and define,

$$D_i^n(\mu||\rho) \equiv \sum_{|x|=i-1} \mu(x) \sum_{|y|=n} \mu(xy|x) \ln \frac{\mu(xy|x)}{\rho(xy|x)}.$$

Thus we can see that,

$$D_1^n(\mu||\rho) = \mu(\lambda) \sum_{|y|=n} \mu(\lambda y|\lambda) \ln \frac{\mu(\lambda y|\lambda)}{\rho(\lambda y|\lambda)} = \sum_{|y|=n} \mu(y) \ln \frac{\mu(y)}{\rho(y)},$$

and so $D_1^1(\mu||\rho) \equiv D(\mu||\rho)$. In a similar fashion we write D^n and D_m for D_1^n and D_m^1

We will require the following two lemmas:

Lemma 1.5.1 *Let μ be a measure and ρ a semi-measure. It follows that*

$$D^n(\mu||\rho) = \sum_{i=1}^{n} D_i(\mu||\rho).$$

Proof. This result follows from the above defintions and a simple application of Bayes' theorem.

$$\begin{aligned}
D^n(\mu||\rho) &= \sum_{|x|=n} \mu(x) \ln \frac{\mu(x)}{\rho(x)} \\
&= \sum_{|x|=n-1} \sum_{|y|=1} \mu(xy) \ln \frac{\mu(xy)}{\rho(xy)} \\
&= \sum_{|x|=n-1} \mu(x) \sum_{|y|=1} \mu(xy|x) \ln \frac{\mu(xy|x)\mu(x)}{\rho(xy|x)\rho(x)}
\end{aligned}$$

$$\begin{aligned}
&= \sum_{|x|=n-1} \mu(x) \ln \frac{\mu(x)}{\rho(x)} \sum_{|y|=1} \mu(xy|x) \\
&\quad + \sum_{|x|=n-1} \mu(x) \sum_{|y|=1} \mu(xy|x) \ln \frac{\mu(xy|x)}{\rho(xy|x)} \\
&= \sum_{|x|=n-1} \mu(x) \ln \frac{\mu(x)}{\rho(x)} + D_n(\mu||\rho) \\
&= D^{n-1}(\mu||\rho) + D_n(\mu||\rho).
\end{aligned}$$

And so by induction on n we obtain the result. $\square$

Lemma 1.5.2 *Let μ and ρ be two probability measures over $\mathcal{A}^*$ where $\mathcal{A} = \{0,1\}$. It follows that for any $x \in \mathcal{A}^*$,*

$$D_{|x|+1}(\mu||\rho) \geq 2(\mu(x0) - \rho(x0))^2.$$

Proof. Let $f(\mu,\rho) = D_{|x|+1}(\mu||\rho) - 2(p-q)^2$ where $p = \mu(x0)$ and $q = \rho(x0)$. Thus,

$$f(\mu,\rho) = p\ln\frac{p}{q} + \ln\frac{1-p}{1-q} - p\ln\frac{1-p}{1-q} - 2(p-q)^2.$$

And so,

$$\begin{aligned}\frac{\partial f}{\partial q} &= 4(p-q) - \frac{p}{q} - \frac{p}{1-q} + \frac{1}{1-q} \\ &= (q-p)\frac{4(q-\frac{1}{2})^2}{q(1-q)}.\end{aligned}$$

Thus the sign of $\partial f/\partial g$ is just the sign of the factor $q-p$ as $q \in [0,1]$ and so q, $(1-q)$ and $(q-\frac{1}{2})^2$ are all positive. If $\mu \equiv \rho$ then $f(\mu,\rho) = 0$, and so for all p and all q we see that $f \geq 0$. That is, $D_{|x|+1}(\mu||\rho) \geq 2(\mu(x0) - \rho(x0))^2$. $\square$

1.6 Recursive Function Theory

Informally an *algorithm* for computing a partial function $\psi : \mathbb{N} \xrightarrow{o} \mathbb{N}$ is a finite set of instructions which, given an input $x \in dom(\psi)$, yields after a finite number $t < \infty$ of steps, the output $y = \psi(x)$. The algorithm must specify unambiguously how to obtain each step in the computation from the previous steps and from the input. We call such a partial function ψ a *partial computable function*. If ψ also belongs to the set of total functions, then *psi* is called a *computable function*. These informal notions have as formal models the *partial recursive functions* and the *recursive functions* respectively. We call any function which is not partial recursive, *non-recursive*.

Perhaps the real significance of these concepts come from a central result in the theory known as *Turing's Thesis*. Informally it tells us that the partial recursive functions are the ones which we could in theory calculate if given a sufficient, but still finite, amount of time, money, people, computers etc. Non-recursive functions on the other hand can't be calculated even with such generous resources avaliable and so in a practical sense aren't all that useful. From this rather simplistic standpoint, we can see that there is an issue of real world practicality at stake here; though it is worth noting that in reality it is only a small subset of even the partial recursive functions which one could ever hope to calculate as the resources avaliable are always limited. For example, while a function which requires a trillion billion supercomputers to calculate is technically speaking still recursive, it certainly isn't very useful in practice.

This is important to us as it is clear that any inductive inference method which is not a recursive function would be of no practical use to anybody. Likewise, any hypothesis learned by an inductive inference method which wasn't a recursive function wouldn't be of much practical use either.

The above notions of computability can be readily extended to cover all sorts of functions with domains and ranges other than $\mathbb{N}$; of particular interest to us are functions over strings and sets of sequences. There are many equivalent ways to approach this topic and far more detailed developments can be found in many texts. For our purposes the following quick overview will suffice.

It is at once clear that given an alphabet and an associated total ordering on its symbols, the functions *string* and $string^{-1}$ are both unambiguous and can be calculated with finite resources. Intuitively then we can see that both of these functions are what we would call recursive. It also seems intuitively clear that the composition of any number of partial recursive functions produces a partial recursive function. Thus the following definition should come as no surprise;

Definition 1.6.1 *A partial function* $\psi : \mathcal{A}^* \xrightarrow{o} \mathcal{A}^*$ *is* partial recursive *if there exits a partial recursive function* $f : \mathbb{N} \xrightarrow{o} \mathbb{N}$ *such that*

$$\forall x \in \mathcal{A}^* \quad \psi(x) = string(f(string^{-1}(x))).$$

Likewise a function $\psi : \mathcal{A}^* \xrightarrow{o} \mathcal{A}^*$ *is* recursive *if there exists a recursive function* $f : \mathbb{N} \to \mathbb{N}$ *such that the above condition holds.*

In a similar fashion other definitions can be developed for functions with multiple arguments and other domains and ranges. The following is a particularly important type of partial recursive function;

Definition 1.6.2 *We call a partial recursive function* $\psi : (\mathcal{A}^*)^n \times \mathbb{N} \to \mathcal{A}^*$ universal *if for all partial recursive functions* $\phi : (\mathcal{A}^*)^n \to \mathcal{A}^*$ *there exists* $i \in \mathbb{N}$ *such that,*

$$\forall x \in (\mathcal{A}^*)^n \quad \psi_i(x) = \phi(x).$$

The cornerstone of the theory is the existence of such functions:

Theorem 1.6.1 *For all* $n \in \mathbb{N}$ *there exists a universal partial recursive function* $\psi : (\mathcal{A}^*)^n \times \mathbb{N} \to \mathcal{A}^*$.

Proof. Omitted. □

Our comment about the composition of partial recursive functions being partial recursive can now be formalised in this context:

Theorem 1.6.2 (Uniform Composition Property) *For a universal partial recursive function* $\psi : (\mathcal{A}^*)^n \times \mathbb{N} \to \mathcal{A}^*$ *there exists a recursive function comp* : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *such that*

$$\forall x \in (\mathcal{A}^*)^n \quad \psi_{comp(i,j)}(x) = \psi_i \circ \psi_j(x).$$

Proof. Ommitted. □

Pick a universal partial recursive function $\psi : \mathcal{A}^* \times \mathbb{N} \to \mathcal{A}^*$ and call the enumeration $\psi_1, \psi_2, \ldots$ the *standard enumeration* of partial recursive functions.

As the function $< \cdot, \cdot > : \mathcal{A}^* \times \mathcal{A}^* \to \mathcal{A}^*$ is bijective and recursive, we can apply the uniform composition property to obtain more general bijective recursive functions $< > : (\mathcal{A}^*)^n \times \mathcal{A}^* \to \mathcal{A}^*$. For example we could define

$< x, y, z > \equiv < x, < y, z >>$ for all $x, y, z \in \mathcal{A}^*$. Thus the number of variables of any partial recursive function can be reduced to one and therefore the standard enumeration $\psi_1, \psi_2, \ldots$ and the function $< \cdot, \cdot >$ are all that is needed for the general theory.

Definition 1.6.3 *A set is* recursively enumerable *if it is empty or the range of a total recursive function. A set is* recursive *if it has a recursive characteristic function.*

Obviously if a set is recursive then it is recursively enumerable.

Now we extend the notion of recursiveness to cover real valued functions and also introduce the weaker properties of enumerability and co-enumerability of functions.

Definition 1.6.4 *A function $f : \mathcal{A}^* \to \mathbb{R}$ is* enumerable *if the set $\{(x, r) \in \mathcal{A}^* \times \mathbb{Q} : r < f(x)\}$ is recursively enumerable. If $-f$ is an enumerable function then we say that f is* co-enumerable*. If f is both enumerable and co-enumerable, we say that f is* recursive*.*

The following lemmas gives us a similar but often more convenient and perhaps more intuitive expressions for enumerable and recursive real valued functions.

Lemma 1.6.1 *A real function $f : \mathcal{A}^* \to \mathbb{R}$ is enumerable iff there exists a recursive function $g : \mathcal{A}^* \times \mathbb{N} \to \mathbb{Q}$ such that for all $x \in \mathcal{A}^*$,*

$$\forall k \in \mathbb{N} \quad g^k(x) \leq g^{k+1}(x)$$

and

$$f(x) = \lim_{k \to \infty} g^k(x).$$

A similar result holds for co-enumerable functions.

Proof. To appear... □

Thus an enumerable function $f : \mathcal{A}^* \to \mathbb{R}$ is one which we can approximate from below while a co-enumerable function is one which we can approximate from above. Trivially we see that any recursive function is enumerable.

Lemma 1.6.2 *A function $f : \mathcal{A}^* \to \mathbb{R}$ is recursive iff there exits a recursive function $g : \mathcal{A}^* \times \mathbb{N} \to \mathbb{Q}$ such that for all $x \in \mathcal{A}^*$,*

$$\forall k \in \mathbb{N} \quad |f(x) - g^k(x)| < \frac{1}{k}.$$

Proof. To appear... □

Thus a recursive function $f : \mathcal{A}^* \to \mathbb{R}$ is one which we can approximate to any specified degree of accuracy.

1.7 Algorithmic Information Theory

Fundamental to algorithmic information theory is a particular type of partial recursive function called a *computer*.

Definition 1.7.1 *A (prefix free or Chaitin)* computer *is a partial recursive function $C : \mathcal{A}^* \times \mathcal{A}^* \xrightarrow{o} \mathcal{A}^*$ such that the set $\{p : d \in \mathcal{A}^*, C(p,d) \neq \infty\}$ is prefix free.*

An intuitive interpretation is to consider p to be a program which the computer C executes and d some data which the program has access to. Of particular importance is the following type of computer:

Definition 1.7.2 *A computer U is* universal *if for each computer C there exists $c \in \mathbb{N}$ depending only on U and C such that whenever $C(p,d) \neq \infty$,*

$$\exists p' \in \mathcal{A}^* \quad U(p',d) = C(p,d)$$

such that

$$|p'| \leq |p| + c.$$

Thus a universal computer is one which we can program to simulate the operation of any other computer. Furthermore, the additional program length needed to simulate any specific computer is of a fixed length and depends only on the universal computer we are using and the computer we wish to simulate.

Theorem 1.7.1 *There effectively exists a universal computer.*

Proof Sketch. This result is a direct consequence of the existence of universal partial recursive functions and the uniform composition property. □

We are now in a position to be able to define a measure of the information content of individual strings.

Definition 1.7.3 *Let C be a computer. The (prefix or Chaitin)* complexity *relative to C of a string $x \in \mathcal{A}^*$ is defined as*

$$H_C(x) \equiv \min\{|p| : p \in \mathcal{A}^*, C(p,\lambda) = x\}.$$

As we will soon prove, this function takes on particular importance when the computer C is a universal computer. Firstly we pick a universal computer $\mathcal{U}$ and call it the *reference computer*.

Definition 1.7.4 *Call the function $H(x) \equiv H_{\mathcal{U}}(x)$ the* complexity function *and for any string $x \in \mathcal{A}$ call $H(x)$ the* complexity *of x.*

The real significance of this function comes from the following useful property:

Theorem 1.7.2 (Invariance Theorem) *For every computer C there exists a constant c depending only on $\mathcal{U}$ and C such that*

$$\forall x \in \mathcal{A}^* \quad H(x) \leq H_C(x) + c.$$

Proof. From the above definitions it immediately follows that for all $x \in \mathcal{A}^*$,

$$\begin{aligned} H_C(x) &= \min\{|p| : p \in \mathcal{A}^*, C(p, \lambda) = x\} \\ &\geq \min\{|p'| - c : p' \in \mathcal{A}^*, \mathcal{U}(p', \lambda) = x\} \\ &= H(x) - c. \end{aligned}$$

And so we have the result. □

Thus we are able to measure the information content of an arbitrary string with a method which is, at least up to a constant, independent of the particular reference machine we have chosen. This gives our measure of information some degree of universality. However this comes at a price as the following theorem shows.

Theorem 1.7.3 *$H(x)$ is not recursive.*

Proof. To appear... □

The fact that the complexity function is not recursive is unfortunate and we will deal with some of its repercussions later on. The weaker condition of co-enumerability does however hold:

Theorem 1.7.4 *$H(x)$ is co-enumerable.*

Proof. To prove that $H(x)$ is co-enumerable we must show that the set $\{(x, r) \in \mathcal{A}^* \times \mathbb{Q} : H(x) < r\}$ is recursively enumerable. This is easy since $H(x) < r$ iff there exists $y \in \mathcal{A}^*$ and $t \in \mathbb{N}$ such that $|y| < n$ and $\mathcal{U}(y, \lambda) = x$ in at most t steps. □

2 Solomonoff Induction

In this chapter we examine Solomonoff's all purpose induction method and prove some impressive results about its performance. Our approach to the topic isn't the exact path the Solomonoff originally used himself; rather we come from the more mathematically general perspective of enumerable semimeasures. We do however try to give the intuitive motivation behind the topic that Solomonoff put forward.

2.1 The General Process of Inductive Learning

Solomonoff's induction method is an attempt to design a general all purpose inductive inference system. Ideally such a system would be able to accurately learn any meaningful hypothesis from a bare minimum of appropriately formatted information. Before trying to define such an inference system and analyse its behaviour, we first need to form a reasonable idea as to what such a system might look like. To help us do this, let's imagine some sort of super intelligent device or being that operates as a perfect inductive inference system. For the sake of our thought exercise we will call this machine or being Zed.

Our question is: What properties will Zed have? Firstly, Zed mustn't be too narrow minded as to what could potentially be a correct hypothesis. It is however clear that any non-computable hypothesis would not be of much use to anybody. Hence restricting Zed's set of possible hypotheses to only the computable ones seems reasonable enough. Exactly what this means will become clear when we formalise all these ideas later on.

Next consider what sort of knowledge Zed has about things before processing any data. For Zed to be truly all purpose, Zed must be able to function in situations where no prior information about the system under investigation is avaliable. For example, in a totally artificial inductive inference problem, even complete knowledge of all the laws of physics would be of no help. This is not to say that Zed shouldn't be able to utilise prior information, but simply that prior information is an extra rather than an essential part of Zed's operation. Thus if we consider Zed's initial state to be independent of the problem at hand, then it follows that this state must be one of complete ignorance about the nature of the system under investigation.

Now that we have some idea about Zed's initial state and the set of potential hypotheses that Zed is going to consider, we next look at what actions Zed will need to be able to perform. Obviously Zed will need to be able to process information in order to determine which hypotheses are likely and which are unlikely or even impossible. Perhaps Zed's first source of such information would be the prior information mentioned above, that is; any knowledge relevant to the system under investigation that comes from work conducted prior to the current investigation. For example, knowledge of the laws of physics would often be helpful when studying real physical systems. It might be the case that others have studied similar systems before. It could even be the case that the correct hypothesis is already known! Thus, just as any serious scientist checks what information already exists before contributing his own ideas and performing his own experiments etc, Zed must at least have the ability to utilise such information when available.

Having exhausted all information deriving from pervious work, the next

avenue must be for Zed to gather further information himself through experimentation and observation. This information will further refine Zed's estimate of the true hypothesis.

It is then possible that the experimental information might point to further areas of previous work that should be taken into account. Perhaps it will indicate new sets of experiments that should be carried out. In this way Zed gathers more and more information, continually updating and refining his degree of belief in the various possible hypotheses. This process is of course the process by which all scientists operate. Various hypotheses gain or lose favour in the light of new information or even old information which has been over looked.

2.2 Solomonoff's Induction Method

Now that we have a vague idea of what an ideal all purpose inductiveinference system might look like, let us now try to put some flesh on these ideas by attempting to define an induction system in Zed's image.

Consider the induction process: Increasingly large amounts of information about experiments, results, known facts etc produce increasingly accurate estimates of the likelihood of the various hypotheses. Our first task is to formalise the way we represent the information from experiments etc. We can do this using the ideas on codes and strings from chapter 1. Perhaps an example is the best explanation.

Imagine that we have a single coin and it is our job to figure out the probability of getting heads. Presumably we would go about this by tossing the coin a large number of times and noting what happened. With an alphabet $\mathcal{A} = \{H, T\}$ we can simply record these results in the form of a sting, for example $HHHTTHTHTHTTTH$. It is not hard to see that all sorts of measured results could be recorded in a similar fashion.

While this is adequate for our simple coin tossing example, in more complex situations we may also need to include information such as the experimental setup used to obtain specific results. For example, imagine that we have three coins numbered 1, 2 and 3 and it is our job to determine which coin is biased. This can simply be done by setting $\mathcal{A} = \{1, 2, 3, H, T\}$ and then recording which coin was tossed followed by the result. Thus the string $1H3T2H2T$ would indicate that we tossed coin 1 first and obtained a head, then we tossed coin 3 and obtained a tail and so on.

As we perform more experiments the data string describing our experiments and results grows longer and longer. Because the number of experiments we can perform is in theory unlimited we can think of our observed data string as being a prefix of some infinitely long sequence. The longer our finite data string grows the smaller the set of sequences with that prefix becomes. Now consider the hypotheses themselves. They are simply statements about the way in which a system behaves, or in this case, statements about various strings which could describe a systems behaviour. Hypotheses which state that the observed data string so far could not have occurred are obviously incorrect hypotheses. On the other hand, a hypothesis which considers the observed data string to be reasonably likely is clearly more likely to be the correct hypothesis.

Thus it is not hard to see that these hypotheses are in fact probability measures over the space $\mathcal{A}^*$ or more correctly, the corresponding probability measures over the measurable space $(\mathcal{A}^\omega, \mathcal{S})$. This gives considerable flexibility

to our hypotheses as it allows us to include both deterministic and stochastic computable processes.

This is perhaps best explained by example. Consider again the situation where we have a single possibly biased coin. If the coin was in fact unbiased the correct hypothesis would be represented by the measure $\mu(x) = 2^{-|x|}$ where x is the data string. This would give equal probability to each string of a given length and thus gives us the correct probability of any observed data string occurring. Clearly this hypothesis is stochastic.

Now consider another hypothesis. If our coin always lands H on even throws and T on odd throws, the correct hypotheses would be represented by the measure

$$\mu(x_1x_2...x_n) = \begin{cases} 1 & \text{if } x_i = H \quad \forall \text{ even } i \\ & \text{and } x_i = T \quad \forall \text{ odd } i, \\ 0 & \text{otherwise.} \end{cases}$$

Thus only strings which consist of alternate H's and T's in even and odd positions respectively have nonzero probability. Obviously then, this hypothesis describes a system that is completely deterministic. In both of these examples, the function μ is clearly computable.

In more realistic induction problems the measure representing the correct hypothesis would be considerably more complex and the data strings would most likely contain prior information and experimental setup details. In fact, we might need to follow some quite complex algorithm that embodies a large amount of information in order to work out the value of the measure for any particalar input string x, but nevertheless, the resources we require are in principle finite and the process well defined — hence the phrase "computable hypothesis".

Given any particular hypothesis, we can then use it to predict future observations by simply conditioning the probability. So in the coin example above with $x = x_1, x_2, \ldots x_n$ being the observed coin tosses so far, the probability that $x_{n+1} = a$ according to a hypothesis would be

$$\mu(xa|x) = \frac{\mu(xa \cap x)}{\mu(x)} = \frac{\mu(xa)}{\mu(x)}.$$

This follows because when viewed as sets, $xa\mathcal{A}^\omega \subset x\mathcal{A}^\omega$. It is worth noting that this is only a prediction based on the assumption that the single hypothesis is the correct one. We now have a flexible all purpose way to represent both our data and hypotheses. What we need next is a method of estimating which hypotheses are likely and which are unlikely or even impossible. Bayes' theorem provides us with such a method, however it demands that we assign prior probabilities to each hypothesis. This is where the real innovation in Solomonoff's technique lies and we will examine his solution and its consequences in the following section.

Before doing so a quick recap is in order: our perfect induction system is going to read in an ever increasing string which contains details of various experiments, their results and other miscellaneous information relevant to the problem. This information will be used to calculate the probability that the various computable hypotheses might be the correct one. Bayes' rule provides us with the mechanism to do this, however it demands that we assign prior probabilities to each hypothesis.

2.3 Solomonoff's Universal Prior

What we are after is a distribution over the set of all hypotheses which does not greatly favour any particular set of hypotheses over any other and thus bias our induction results. Trivially we can see that the set of hypotheses is infinite and so simply assigning each hypothesis to have an equal prior probability is mathematically impossible. Thus necessarily some hypotheses will have higher prior probabilities than others. Solomonoff's solution was to devise what could be considered the natural distribution over the set of computable hypotheses from the perspective of computability theory.

From the previous section, each hypothesis is represented by an enumerable semi-measure. From the definition of an enumerable function we know that each of these enumerable semi-measures has in turn at least one partial recursive function or equivalently a computer which can be used to compute it. Thus any distribution over the set of computers induces a corresponding distribution over the set of all acceptable hypotheses. Fortunately, we can determine quite a natural prior distribution over the set of computers with the aid of our universal reference computer $\mathcal{U}$ because each computer is represented by at least one program for $\mathcal{U}$ and a good uninformative prior distribution over the set of all programs can simply be generated by using an unbiased dice to produce successive digits from our alphabet. These random digits can then be fed into $\mathcal{U}$ as a program specifying which computer to emulate. As $\mathcal{U}$ is a prefix machine, $\mathcal{U}$ will be able to detect the end of the program automatically and then execute it. So we are in a sense, picking programs or equivalently hypotheses at random. Thus we have derived a distribution over the set of permissable hypotheses form nothing more than a simple uniform distribution over $\mathcal{A}^\omega$ and basic computability theory.

Let us now formalise these ideas. Let μ be some enumerable semi-measure representing a hypothesis. The universal prior probability of this hypothesis is defined to be

$$P^U(\mu) \equiv \sum_{\mathcal{U}(p,x)=\mu(x)} Q^{-|p|},$$

where $\mathcal{U}(p,x) = \mu$ means that the program p causes $\mathcal{U}$ to calculate the enumerable semi-measure μ on data x. Q is the number of symbols in our alphabet.

It would appear that our distribution over all permissible hypotheses can contain no significant information other than that contained in our choice of universal reference computer $\mathcal{U}$. By the invariance theorem and the so called "coding theorems" (which have not been presented here) the affect of this choice is restricted. Another tactic is to use universal computers which are very simple, and thus contain little information. Discussion of these topics is left for more advanced texts.

In the above definition of P^U we have summed over all programs p which describe the semi-measure μ. However it is clear that only the shortest program for will have much affect on $P^U(x)$. So we can approximate P^U by

$$P(\mu) \equiv Q^{-H(\mu)},$$

where $H(\mu)$ is defined to be the length shortest program that computes μ. Various results exist in the literature detailing precise bounds on this approximation which we will not explore.

Now consider again the problem of predicting the continuation of the data sequence. The above prior distribution induces a new distribution over $\mathcal{A}^*$ when we take all the hypotheses into account; we simply take a sum over all hypotheses of the prior probability of each hypothesis times the probability the hypothesis gives to the observed data string. Thus we define,

$$\mathbf{M}(x) \equiv \sum_{\mu \in \mathcal{M}^R} Q^{-H(\mu)} \mu(x),$$

where $\mathcal{M}^R$ is the set of all recursive semi-measures, that is, that the set of all our computable hypotheses.

Thus our best possible prediction of the continuation of a data string x taking all possible hypotheses into account is now

$$\mathbf{M}(xa|x) = \frac{\mathbf{M}(xa)}{\mathbf{M}(x)}.$$

2.4 Dominant Enumerable Semi-Measures

The semi-measure $\mathbf{M}$ is actually an example of what we call a *dominant enumerable semi-measure*. This property of being dominant gives $\mathbf{M}$ very powerful properties as a prior distribution. In this section we define and prove the existence of a general class of dominant enumerable semi-measures and show that $\mathbf{M}$ belongs to this class.

Let $\mathcal{M}$ be the class of all enumerable semi-measures over $\mathcal{A}^*$ and let $\mathcal{M}^R$ be the class of all recursive semi-measures. Thus we see that $\mathcal{M}^R \subset \mathcal{M}$.

Definition 2.4.1 *A semi-measure $\mu \in \mathcal{M}$ is* dominant *over $\mathcal{M}$ iff for all $\rho \in \mathcal{M}$, there exists $c > 0$ such that*

$$\forall x \in \mathcal{A}^* \quad \mu(x) \geq c\rho(x).$$

It is clear from the above definition that a measure which is dominant must have its probability mass spread very thinly over the space and so in some sense will contain very little information. Thus it appears reasonable that a dominant semi-measure might be useful as a non-informative universal prior distribution for inference purposes.

Before we can prove the enumerability of $\mathcal{M}$ we must first prove the following:

Lemma 2.4.1 *The class of enumerable functions of the form $\mathcal{A}^* \to \mathbb{R}^+$ is recursively enumerable.*

Proof. By lemma 1.6.1 we see that a function is enumerable if and only if it has a ration valued recursive function approximating it from below. Thus the idea of this proof it to construct the desired enumeration of all enumerable functions by creating an enumeration of all rational approximation functions.

Let $\psi_1, \psi_2, \ldots$ be the standard recursive enumeration of all partial recursive functions of the form $\mathcal{A}^* \to \mathcal{A}^*$ and define $\phi_i : \mathcal{A}^* \times \mathbb{N} \to \mathcal{A}^*$ to be

$$\phi_i(x, k) \equiv \psi_i(< x.string(k)) >).$$

Clearly $\phi_1, \phi_2, \ldots$ is a recursive enumeration. As $< \cdot, \cdot >$, *string* and each ψ_i is partial recursive, each ϕ_i will also be partial recursive by the uniform

composition property. Because $< \cdot, \cdot >$ is bijective, the enumeration $\phi_1, \phi_2, \ldots$ contains all partial recursive functions of this form.

Now define $f_i : \mathcal{A}^* \times \mathbb{N} \to \mathbb{Q}$ to be

$$f_i(x, j) \equiv \frac{m}{n},$$

where $\psi_i(x, j) =< string^{-1}(m), string^{-1}(n) >$. By a similar argument to that used above, we can see that $f_1, f_2, \ldots$ is a recursive enumeration of all partial recursive functions of this form.

We need each approximation function to be recursive, not just partial recursive. The following algorithm produces the desired sequences of recursive functions:

Step 1 : Set $s := 0$, $p := 0$ and $h_i^0(x) := -\infty$ for all $x \in \mathcal{A}^*$

Step 2 : Do one step in the calculation of $f_i(x, p)$
Let $s := s + 1$

Step 3 : If the calculation of $f_i(x, p)$ has finished
let $p := p + 1$
let $h_i^s(x) := f_i(x, p)$
otherwise
let $h_i^s(x) := h_i^{s-1}(x)$

Step 4 : Go to step 2

Clearly steps 1, 3 and 4 can be done with finite resources. As step 2 only carries out one step in the calculation of $f_i(x, p)$, this must also be acceptable — even if the calculation of $f_i(x, p)$ is never going to terminate. Thus for each i this algorithm produces a sequence of rational valued recursive functions $h_i^0, h_i^1, \ldots$.

We now modify these functions slightly to form rational approximation functions $g_i^k : \mathcal{A} \to \mathbb{Q}$. Define

$$g_i^k(x) \equiv \max_{j \leq k} h_i^j(x).$$

By lemma 1.6.1 we see that any enumerable function must have a monotonically increasing rational valued recursive function approximating it from below, thus the recursive enumeration defined by

$$g_i(x) \equiv \lim_{k \to \infty} g_i^k(x)$$

must contain all and only enumerable functions. □

It is worth noting that the operation of taking a limit to infinity is nonrecursive and so g_i isn't necessarily a partial recursive function. Indeed as we have seen previously, the set of partial recursive functions is a only subset of the enumerable functions.

Now that we have created a recursive enumeration of all enumerable functions we can now take this one step further and create a recursive enumeration of all enumerable semi-measures. Essentially we do this by specifying an algorithm which changes the enumeration $g_1, g_2, \ldots$ into a new recursive enumeration of enumerable functions $\rho_1, \rho_2, \ldots$ such a way as to insure that each ρ_i is a semi-measure.

Lemma 2.4.2 *The class of enumerable semi-measures $\mathcal{M}$ is recursively enumerable.*

Proof. Let $g_1, g_2, \ldots$ be the recursive enumeration of all enumerable functions and $g_1^k, g_2^k, \ldots$ the associated rational recursive approximation functions. To obtain a recursive enumeration of all enumerable semi-measures we apply the following algorithm to each g_i:

Step 1 : Set $k := 0$ and $\rho_i(x) := 0$ for all $x \in \mathcal{A}^*$

Step 2 : Set $k := k + 1$

Step 3 : Compute $g_i^k(x)$ for all $x \in \{y \in \mathcal{A}^* : |y| \leq k\}$

Step 4 : If either $g_i^k(\lambda) \geq 1$ or

$$\exists x \in \{y \in \mathcal{A}^* : |y| \leq k-1\} \qquad g_i^k(x) \leq \textstyle\sum_{|a|=1} g_i^k(xa)$$

then stop

Step 5 : Set $\rho_i(x) := g_i^k(x)$ for all $x \in \{y \in \mathcal{A}^* : |y| \leq k\}$

Step 6 : Go to step 2

Clearly steps 2 and 6 are unambiguous and require only finite resources. A function which is identically zero is trivially recursive and so step 1 is acceptable. As the function g_i^k is recursive and the set $\{y \in \mathcal{A}^* : |y| \leq k\}$ finite, steps 3 and 5 are also acceptable. Similarly the sum in step 4 is always finite. Perhaps the key point to notice about this algorithm is that before we update the approximation ρ_i we first check that the new approximation will still be a semi-measure. Thus at all times ρ_i is a semi-measure.

Now consider the two possible ways that the algorithm can go: The first possibility is that g_i isn't a semi-measure. This will at some stage be picked up in step 4, ending the approximation process. As noted above, ρ_i will be a semi-measure when we stop. Indeed, it will even be recursive as we can calculate its value in finitely many steps.

On the other hand; if the function g_i is a semi-measure then we simply continue to approximate the enumerable semi-measure from below forever. Thus if g_i is a semi-measure, $\rho_i = g_i$. In particular this means that the new enumeration $\rho_1, \rho_2, \ldots$ will contain all enumerable semi-measures as the enumeration $g_1, g_2, \ldots$ already contains all enumerable functions and thus all enumerable semi-measures. □

We can now prove the following:

Theorem 2.4.1 *There exists dominant semi-measures over $\mathcal{M}$.*

Proof. Let $\rho_1, \rho_2, \ldots$ be the recursive enumeration of enumerable semi-measures in lemma 2.4.2 and let $\alpha : \mathbb{N} \to \mathbb{R}^+$ be any enumerable function such that

$$\sum_{n=1}^{\infty} \alpha(n) \geq 1.$$

Now define $\nu : \mathcal{A}^* \to \mathbb{R}^+$ as

$$\nu(x) \equiv \sum_{n=1}^{\infty} \alpha(n)\rho_n(x).$$

Firstly we will establish that $\nu \in \mathcal{M}$, then we will prove that ν is in fact dominant over $\mathcal{M}$. As each ρ_n is a semi-measure it is immediately clear that

$$\nu(\lambda) = \sum_{n=1}^{\infty} \alpha(n)\rho_n(\lambda) \leq \sum_{n=1}^{\infty} \alpha(n) \leq 1,$$

and for all $x \in \mathcal{A}^*$,

$$\begin{aligned}
\nu(x) &= \sum_{n=1}^{\infty} \alpha(n)\rho_n(x) \\
&\geq \sum_{n=1}^{\infty} \alpha(n) \left(\sum_{|a|=1} \rho_n(xa) \right) \\
&= \sum_{|a|=1} \sum_{n=1}^{\infty} \alpha(n)\rho_n(xa) \\
&= \sum_{|a|=1} \nu(xa).
\end{aligned}$$

Thus ν is a semi-measure on $\mathcal{A}^*$.

As each ρ_n is enumerable, by lemma 1.6.1 there exists a recursive function ρ_n^k such that $\lim_{k\to\infty} \rho_n^k(x) = \rho_n(x)$ with $\rho_n^k(x) \leq \rho_n^{k+1}(x)$. Likewise we can define a recursive function $\alpha^k(n)$ as $\alpha(n)$ is also an enumerable function. Now define

$$\nu^k(x) \equiv \sum_{n=1}^{k} \alpha^k(n)\rho_n^k(x).$$

Immediately we see that ν_k is increasing in k and $\lim_{k\to\infty} \nu^k(x) = \nu(x)$. As α^k, ρ_n^k and the operation of multiplication and finite summation are recursive, by the uniform composition property ν^k is also recursive. Thus by lemma 1.6.1 again, we see that ν is an enumerable function. Hence $\nu \in \mathcal{M}$.

Finally if $\rho_m \in \mathcal{M}$ then we see that for all $x \in \mathcal{A}^*$,

$$\begin{aligned}
\nu(x) &= \sum_{n=1}^{\infty} \alpha(n)\rho_n(x) \\
&\geq \alpha(m)\rho_m(x).
\end{aligned}$$

Thus ν is dominant over $\mathcal{M}$. □

We have now proven the existence of not just one dominant enumerable semi-measure but of a whole set of such functions. In particular, we are able to choose any function α satisfying the given constants. As it turns out, all dominant measures make very good universal prior distributions. Nevertheless, certain

dominant measures make more intuitive sense than others; indeed Solomonoff did not originally approach this from the perspective of dominant measures, but rather he was looking for a sensible distribution over the set of all computable hypotheses and in the process he founded algorithmic information theory.

Theorem 2.4.2 $\mathbf{M}$ *is a dominant enumerable semi-measure.*

Proof. From theorem 1.7.4 we know that H is co-enumerable and so it follows that Q^{-H} is enumerable. As μ is also enumerable, $\mathbf{M}$ must be enumerable. Trivially we see that $\mathbf{M}$ is also a semi-measure. Because $\mathcal{U}$ is a prefix free universal computer, the set of programs describing the enumerable semi-measures $\mu \in \mathbf{M}$ must be prefix free. Thus by Kraft's Inequality (theorem 1.2.1) we see that P satisfies the conditions on α in theorem 2.4.1 and so $\mathbf{M}$ must be a dominate enumerable semi-measure. □

This gives us the following simple result which we will need in the next section.

Corollary 2.4.1 *For any semi-measure* $\mu \in \mathcal{M}$*,*

$$(\ln Q)H(\mu) \geq \ln \frac{\mu(x)}{\mathbf{M}(x)}.$$

Proof. Because $\mathbf{M}$ is dominant, for any enumerable semi-measure $\mu \in \mathbf{M}$ it must be the case that $\mathbf{M}(x) \geq Q^{-H(\mu)}\mu(x)$. The result then follows. □

2.5 Completeness of Solomonoff Induction

At last we are able to prove an important result which shows that the error in prediction when using $\mathbf{M}$ instead of the true recursive prior μ always diminishes to zero very rapidly irrespective of what the unknown μ might be. To simplify our analysis let us assume that the data sequence with experimental details, results etc. has been encoded as a binary sequence, that is, let $\mathcal{A} = \{0, 1\}$. Doing so gives an added symmetry to prediction errors in that it makes the error in probability in predicting the nth digit to be a 0 the same as for predicting a 1. Thus the error in probability in the nth prediction is in both cases simply represented by

$$|\mathbf{M}(x0|x) - \mu(x0|x)|,$$

where μ is the true recursive prior distribution.

We are not so interested in the error in prediction for any one data sequence but rather we are more interested in the general behaviour which is expressed by the average or expected error in the nth prediction taken over the set of all possible data sequences. For our analysis it will be more convenient to work with the square error in the nth prediction rather than the absolute error. Hence we get

$$S_i \equiv \sum_{|x|=i-1} \mu(x)(\mathbf{M}(x0|x) - \mu(x0|x))^2,$$

which is the expected squared error in the nth prediction.

Theorem 2.5.1 *Let μ be a recursive measure and* $\mathbf{M}$ *our dominant enumerable semi-measure. It follows that*

$$\sum_{i=1}^{\infty} S_i \leq \frac{\ln 2}{2} H(\mu).$$

Proof. By the definition of D^n and corollary 2.4.1 we see that for all $n \in \mathbb{N}$,

$$\begin{aligned} D^n(\mu||\mathbf{M}) &= \sum_{|x|=n} \mu(x) \ln \frac{\mu(x)}{\mathbf{M}(\mathbf{x})} \\ &\leq (\ln 2) H(\mu) \sum_{|x|=n} \mu(x) \\ &= (\ln 2) H(\mu). \end{aligned}$$

Because this holds for all $n \in \mathbb{N}$, it follows from lemma 1.5.1 that,

$$\sum_{i=1}^{\infty} D_i(\mu||\mathbf{M}) \leq (\ln 2) H(\mu).$$

Before preceeding further we need to make $\mathbf{M}$ into a proper probability measure. We do this by the method outlined in chapter 2. Let $\mathcal{A}_u = \mathcal{A} \cup \{u\}$ and define a probability measure $\mathbf{M}'$ over $\mathcal{A}_u^*$ by

$$\begin{aligned} \mathbf{M}'(x0|x) &\equiv \mathbf{M}(x0|x) \\ \mathbf{M}'(x1|x) &\equiv 1 - \mathbf{M}(x0|x). \end{aligned}$$

Further, extend the probability measure μ over $\mathcal{A}_u^*$ by defining $\mu(x) \equiv 0$ for all $x \notin \mathcal{A}^*$.

From the definition of D_i it now follows that,

$$\sum_{i=1}^{\infty} D_i(\mu||\mathbf{M}') \leq \sum_{i=1}^{\infty} D_i(\mu||\mathbf{M}) \leq (\ln 2) H(\mu).$$

By lemma 1.5.2 we see that,

$$\begin{aligned} D_i(\mu||\mathbf{M}') &= \sum_{|x|=i-1} \mu(x) \sum_{|y|=1} \mu(xy|x) \ln \frac{\mu(xy|x)}{\mathbf{M}'(xy|x)} \\ &\geq 2 \sum_{|x|=i-1} \mu(x)(\mu(x0|x) - \mathbf{M}'(x0|x))^2 \\ &= 2S_i. \end{aligned}$$

Thus we obtain the result;

$$\sum_{i=1}^{\infty} S_i \leq \frac{ln2}{2} H(\mu).$$

□

Because the harmonic series $\sum \frac{1}{n}$ does not converge, while S_i on the other hand does converge by the previous theorem, it follows that the sequence S_i must converge to zero faster than $\frac{1}{n}$. In other words; when using our universal prior $\mathbf{M}$ for prediction, the error in the nth prediction goes to zero faster than $\frac{1}{n}$ *irrespective* of what the unknown true prior might be! Thus we have defined a very powerful system for prediction and inductive inference. It is easy to see that a similar result to theorem 2.4.1 can be proven for other dominant enumerable semi-measures. This allows us to prove a similar result to the above theorem for other dominant enumerable semi-measures, hence the reason for considering Solomonoff's induction method to be a special case of a general class of powerful inductive inference methods which use dominant enumerable priors.

2.6 Properties of Dominant Measures

Unfortunately there is a catch, and a very serious one for anybody wishing to use Solomonoff's inductive method in practice. Firstly we need two lemmas.

Lemma 2.6.1 *If an enumerable semi-measure is a probability measure then it is recursive.*

Proof. Let μ be an enumerable probability measure. If we can prove that μ is also co-enumerable then we will have proven that μ is recursive. By lemma 1.6.1 there exists a rational valued recursive function $g^k(x)$ increasing in k such that $\lim_{k\to\infty} g^k(x) = \mu(x)$ for all $x \in \mathcal{A}^*$. Define $C(x) \equiv \mathcal{A}^{|x|} \setminus x$ and

$$h^k(x) \equiv \sum_{y \in C(x)} g^k(y) - 1.$$

Clearly h^k is a rational valued recursive function and is increasing in k. Because the set $C(x) \cup x = \mathcal{A}^{|x|}$ partitions the space $\mathcal{A}^\omega$ and μ is a probability measure, it follows that

$$\begin{aligned}
\lim_{k\to\infty} h^k(x) &= \sum_{y \in C(x)} \lim_{k\to\infty} g^k(y) - 1 \\
&= \sum_{y \in C(x)} \mu(y) - 1 \\
&= (1 - \mu(x)) - 1 \\
&= -\mu(x).
\end{aligned}$$

Thus by lemma 1.6.1 we see that $-\mu$ is enumerable; that is, μ is co-enumerable. As it is both enumerable and co-enumerable it must be recursive. □

Lemma 2.6.2 *The class $\mathcal{M}^R$ has no dominant semi-measure.*

Proof. To appear... □

Now we can prove the desired (?!) result:

Theorem 2.6.1 *A measure dominant over $\mathcal{M}$ cannot be a probability measure or recursive.*

Proof. Let ν be a semi-measure which is dominant over $\mathcal{M}$. Clearly ν cannot be recursive because this would make ν dominant over $\mathcal{M}^R$ contradicting the previous lemma. As ν is an enumerable semi-measure and not recursive, ν isn't a probability measure by lemma 2.6.1. □

This result appears to be fatal to any hopes of building a Bayesian inductive inference system which employs a dominant distribution as a prior; firstly any such distribution would not be a probability measure and secondly we wouldn't be able to compute the distribution anyway! Thus it is clear that any such induction system could not itself be of any direct practical use. However, inductive inference principles such as Occam's razor, the maximum likelihood principle and the minimum description length principle can all be seen as computable approximations to Solomonoff's perfect but uncomputable inductive inference method. Hence the theoretical interest in inference methods such as Solomonoff's is well justified, even if it is only to aid those pursuing more effective computable approximations for practical purposes and for providing a unifying perspective on the many diverse principles and techniques used in inductive inference.

References

[1] Cristian S. Calude. *Information and Randomness: an algorithmic perspective.* Springer-Verlag, 1994.

[2] Ming Li and Paul M. B. Vitányi. Inductive Reasoning and Kolmogorov Complexity. *Journal of Computer and System Sciences*, 44:343-384, 1992.

[3] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, 1997.

[4] Ray J. Solomonoff. A formal theory of inductive inference, Part 1 and Part 2, *Inform. and Control* 7:1-22 and 224-254, 1964.

[5] Ray J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems, *IEEE Trans.* IT-24:422-432, 1978.

[6] A. K. Zvonkin and Leonid A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms, *Russian Math. Survey.* 25(6):83-124, 1970.