A Syntactic Approach to Prediction

John Woodward, Jerry Swan

Computer Science, The University of Nottingham UK.
{john.woodward,
jerry.swan}
@nottingham.ac.uk
http://www.cs.nott.ac.uk/~jrw,~jps

Abstract. A central question in the empirical sciences is; given a body of data how do we best attempt to make predictions? There are subtle differences between current approaches which include Minimum Message Length (MML) and Solomonoff's theory of induction [24].
The nature of hypothesis spaces is explored and we observe a correlation between the complexity of a function and the frequency with which it is represented. There is not a *single* best hypothesis, as suggested by Occam's razor (which says prefer the simplest), but *a set* of functionally equivalent hypotheses. One set of hypotheses is preferred over another set because it is larger, thus giving the impression simpler functions generalize better. The probabilistic weighting of one set of hypotheses is given by the relative size of its equivalence class. We justify Occam's razor by a counting argument over the hypothesis space.
Occam's razor contrasts with the No Free Lunch theorems which state that it impossible for one machine learning algorithm to generalize better than any other. No Free Lunch theorems assume a distribution over functions, whereas Occam's razor assumes a distribution over programs.

Keywords:
Bias, Conservation of Generalization, Induction, No Free Lunch (NFL), Occam's Razor, Complexity, Machine Learning, Genetic Programming.

1 Introduction

There are at least three approaches to the problem of induction; (1) compression (Kolmogorov Complexity), (2) probability (Algorithmic Complexity), and (3) minimum message length (Information Theory), each with similarities and subtle differences (see sections 1 of [4, 24]). Turing machines feature in the first two, while a coding scheme features in the third. (1) and (3) look for an explanation (i.e. a single theory), while (2) takes a weighted sum over all hypotheses in order to make predictions (see section 8 of [24] and section 4 of [4]). While Solomonoff [22, 23] laid the foundations for a predictive approach, and recently emphasized genetic programming [21]. One of the reasons being that it can embrace recursion, unlike some other methods. In addition, genetic programming readily lends itself to both explanation and prediction motives, in that genetic programming can either return a single program, or can make predictions based on the current population (so called ensemble methods). This paper examines induction from a genetic programming perspective.

A hypothesis is an effective procedure mapping input data to output data. Induction is the process of being given a set of input–output pairs, finding a hypothesis which makes accurate predictions [2, 13, 14, 22, 23]. The process of generating hypotheses is largely done by scientists, but is now the focus of machine learning [14]. This is realized by defining a hypothesis space (program space or search space). A hypothesis space is the set of all instances of some representation (with some size limit imposed). It is then sampled, often using nature inspired methods, to find a hypothesis. The subject of this article is about the static nature of hypothesis spaces, rather than the dynamics of how they are sampled. Investigations into the nature of computational hypothesis spaces [10, 12] leads to two key observations;

KEY OBSERVATION 1: Simple functions are represented more frequently in a hypothesis space, and complex functions are represented less frequently.

KEY OBSERVATION 2: Above a certain size of program, the frequency distribution of functions represented in a hypothesis space does not alter.

Occam's razor has been adopted by the machine learning community and has been taken to mean; "*The simplest explanation is best"* [2], echoed BY Einstein; "Make everything as simple as possible, but no simpler". It has been argued that simpler explanations are more likely to give better predictions on unobserved data. Kearns et. al. [9] state that Occam's razor has become a central doctrine of scientific methodology. Occam's razor is essentially a meta-hypothesis; it is a hypothesis about hypotheses. Why is a simpler hypothesis more likely to generalize to unseen data than a more complex hypothesis? Webb [25] states; "*Several attempts have been made to provide theoretical support for the principle of Occam's razor in the machine learning context. However, these amount to no more than proofs that there are few simple hypotheses and that the probability that one of any such small selection of hypotheses will fit the data is low*". We also criticize these attempts in Section 3. While Webb's paper [25] suggests difficulties with Occam's Razor, Needham and Dowe [16] do cast doubt over this and suggest further investigation in needed.

The NFL theorems [20, 26, 27] state that over all functions, no learning algorithm has superior performance, negating Occam's razor [3, 8]. We should therefore scrutinize the assumptions. NFL is argued in semantic terms (i.e. distributions over functions). We argue in syntactic terms (i.e. distributions over programs) to support Occam's Razor.

The remainder of this article is as follows; In Section 2, examples are given where multiple explanations of the observations are plausible, illustrating that hypotheses should only be falsified if they are shown to be incorrect. In Section 3, previous arguments for parsimony are examined. In Section 4, we examine the

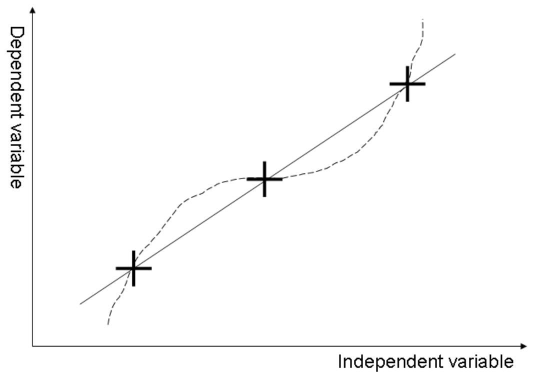

Fig. 1. What is the underlying trend which best explains the data? Two possibilities are shown here, a straight line and a curved line. An Occam's razor approach would prefer a simple straight line, compared to a more complicated curvy line, assuming that the straight line is simpler than the curved line. A NFL approach would say both lines are equally good descriptions of the observation, as all functions have equal probability. We will argue that the more complex wavy line is *a possible explanation* of the data, but with *a lower probability* than the simpler straight line.

assumptions behind NFL theorems. In Section 5, formal definitions are given. Section 6 explores the nature of hypothesis spaces. In Section 7, Occam's razor is proved based on assumptions which have empirical support. The final Section summarizes this article.

2 Illustrative Examples

We give examples in which alternative hypotheses are equally consistent with observation. While a simpler hypothesis may be consistent with the data, we can never discard the possibility that a more complex hypothesis is isomorphic to the process which is responsible for what we observe. Empirically we cannot differentiate two hypotheses which are functionally equivalent on the observations. An experiment can never be constructed which distinguishes them. We can only make statements about the validity of a hypothesis at a semantic level, (i.e. regarding the predictions about input and output).

Imagine we have a computer program which generates the sequence ($<$ $input, output >$ pairs) $< 1, 1 >, < 2, 2 >, < 3, 3 >$ and we are given the task of finding a function which describes the data. Given the data the following three hypothesis are all valid explanations;

$$\begin{aligned}
&H1 : y = x\\
&H2 : y = x + e^{i\pi} + sin^2(x) + cos^2(x)\\
&H3 : y = 0 \text{ if } x = 100 \text{ else } y = x
\end{aligned}$$

All three hypotheses are syntactically different. $H1$ is semantically equivalent to $H2$, and therefore make identical predictions (like Newton and Lagrange, see following paragraph). However $H1$ and $H2$ are semantically distinct from $H3$, which is observationally identical to $H1$ and $H2$ except at the point $x = 100$. The measurement at the point $x = 100$, and only this measurement, will allow us to distinguish between these hypotheses and discard either $H3$ or $H1$ and $H2$. No measurement will ever distinguish between $H1$ and $H2$. While $H1$ and $H2$ are trivially semantically equivalent, it maybe the case that a computer program equivalent to $H2$ is responsible for generating the observed data. We are not interested in inducing a program syntactically equivalent to the program responsible producing the data. We are interested in inducing a program semantically equivalent to the program producing the data. Note that in example, as in this paper, we have ignored noise.

Let us now take an example from physics. Newton's 2nd law of motion is a perfectly good description of "everyday mechanics", where the path of particles is determined by forces. An alternative formulation is Lagrangian mechanics, which determines the path a particle takes by minimizing a quantity based on kinetic and potential energies (i.e. the action functional is stationary). These two approaches are formally equivalent (while a Relativistic formulation of mechanics *is* different). Based on experimental observation, we cannot differentiate between these two theories. A similar situation exists in optics. Indeed (ignoring quantum mechanics and relativity for a moment), assuming the universe is a physical computer (i.e. hardware), we will never know if the software running on it is a Newtonian or Lagrangian program, we can only make observations which will never distinguish between them. Newton and Lagrange are analogous to $H1$ and $H2$ in the preceding paragraph.

3 Justifications and Criticisms of Occam's Razor

[14, 19] provide similar arguments. They essentially reduce Occam's razor to the fact that there are fewer shorter hypotheses than longer ones. [19] states: "*There are far fewer simple hypotheses than complex ones, so that there is only a small chance that* any *simple hypothesis that is wildly incorrect will be consistent with all observations*". A simple hypothesis has fewer degrees of freedom, so if it is consistent with the training data, it is more likely to be consistent with validation data when compared to a more complex hypothesis.

[14] states; "*There are fewer short hypotheses than long ones (based on straightforward combinatorial arguments), it is less likely that one will find a short*

hypothesis that coincidentally fits the training data. In contrast there are often many very complex hypotheses that fit the current training data but fail to generalize correctly to subsequent data". While this is true, it then raises the question that, while there are also many complex hypotheses that fit the data but will fail to generalize, there are also many complex hypotheses that do fit subsequent data. It still does not explain why we should choose a shorter hypothesis in preference to another one. We should not arbitrarily discard more complex hypotheses which are consistent with the data. For example, in Section 2, we should not discard Lagrange over Newton on grounds of complexity.

There has also been empirical work for and against Occam's Razor. [3, 15, 25]. However, further investigations has revealed flaws in the experimental approach. For example, in the induction of decision tree, node cardinality was used as an interpretation of Occam's Razor [15]. However, the insightful work of Needham and Dowe, suggests a more sophisticated Minimum Message Length approach is needed (also see [4] section 4). And, as the title suggests from a paper, *"very simple classification rules perform well on most commonly used datasets"* [6].

4 No Free Lunch Theorems and Generalization

The NFL theorems are a collection of theorems, which broadly state that, any gain in performance regarding generalization on one problem, is reflected by a loss in performance on another problem, and clearly contradict Occam's Razor. These are referred to as NFL theorems [26, 27], the law of Conservation of Generalization [20] and the Ugly Duckling theorems [5]. We will refer to these collectively as NFL as this is the more established term. Over a set of all problems (functions), the positive performance over some learning situations must be offset by an equal degree of negative performance in others, where the performance of a learning algorithm is measured in terms of its prediction of cases not included in the training set. For every problem a learner performs well on, there exists a problem on which the learner performs poorly. An alternative but equivalent statement is, generalization is a zero sum game. There are various ways of stating these theorems, but they are all essentially equivalent as one can be proved from another. All of these arguments for NFL are in purely semantic terms. That is, they are trying to prove a proposition about the distribution over function (semantics), by making assumptions about the probability distribution over functions (semantics). In section 7 we will assume a distribution over programs to prove Occam's razor.

5 Definitions

Definition 1 (Function). *A function is a mapping from one set to another.*

Definition 2 (Program). *A program is an implementation of a function.*

The semantic concept of "increment" is expressed in many different ways by a number of syntactically correct programs containing symbols from the set

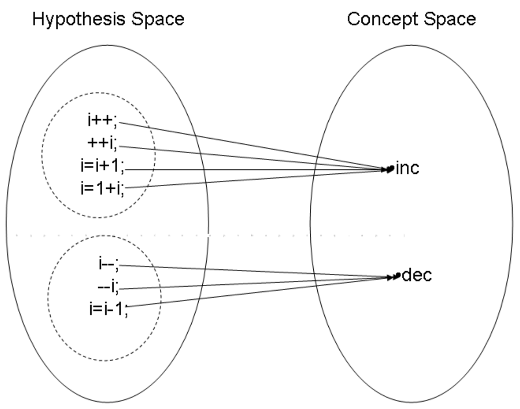

Fig. 2. The space of hypotheses (syntactic structures) maps to the space of concepts (semantic entities). For example, a set of programs maps to a set of functions. Many hypotheses can map to the same concept. The hypothesis space is partitioned into equivalence classes defined by the concepts they map onto. Two equivalence classes are shown by dotted ellipses. Programs representing the same function belong to the same equivalence class.

$\{x, +, 1, ;\}$, see figure 2. One semantic entity (function) can be expressed by many syntactic structures (programs). Functions correspond to semantic entities, and programs correspond to syntactic structures. A hypothesis corresponds to a program and a set of predictions corresponds to a function.

Definition 3 (Set of Base Functions). *A set of base functions (or primitives) is a finite set of functions $\{f_1, \ldots, f_n\}$ which can be used to generate new functions (e.g. a function h is constructed from f_1 and f_2; $h(x) = f_1(f_2(x))$). We are not concerned with how the base functions are implemented, which can be thought of as atomic black boxes. The set of base functions is the same as functions included in the function set of Genetic Programming, where the goal is to synthesize a target function from the base set [10].*

Langdon [12]) uses different sets of base functions for example {NAND}, {AND, OR, NAND, NOR}, and $\{+, -, *, \%\}$, see Section 6 of this paper. Note that none of these sets are capable of expressing the computable functions.

Definition 4 (Size). *The size of a program is the total number of bits needed to express it. (However, we could use any reasonable definition of size. For example, in [12], the size is the number of nodes in the parse tree.)*

Definition 5 (Descriptive Complexity). *The descriptive complexity of a function, $c(f)$, is the size of the smallest program which expresses it, with respect to a set of base functions and data structure. This is the same definition as often used in Genetic Programming [28, 29].*

If the set of base functions is capable of expressing the computable functions, and the definition of size is the number of bits, then this definition is equivalent to Kolmogorov Complexity [13]. The descriptive complexity of a function will in general depend on the set of base functions.

Definition 6 (Equivalence Class of Hypotheses). *An equivalence class of hypotheses contains all hypotheses mapping to the same function.*

For example, a hypothesis space of Java programs maps to a concept space of computable functions. This mapping is achieved by the Java Virtual Machine ($I(p) = f$, where p is a Java program, f is a computable function and I is the Java Virtual Machine). $I(p_i) = I(p_j)$ implies the programs p_i and p_j compute the same function and are semantically equivalent, while $p_i \neq p_j$ implies the programs are syntactically different. A hypothesis space is a set of programs and a concept space is a set of functions.

The hypothesis space is partitioned into equivalence classes of semantically equivalent hypotheses. For example the Java/C programs $\{y = x + 1;\}, \{y = 1 + x;\}, \{y = x + +;\}$, and $\{y = + + x;\}$ all belong to the same class called increment (figure 2). Newtonian and Lagrangian formulations of mechanics belong to the same equivalence class (called classical mechanics), while Special relativity falls into different class as it makes different predictions. During the process of induction, we are eliminating functions inconsistent with the data, and therefore discarding whole equivalence classes (sets of programs) which compute these functions. This leaves us with a set of equivalence classes, all of which are consistent with the data. We are going to argue that the largest equivalence class of this set is the most probable to contain the hypothesis which generalizes best.

Definition 7 (Bias). *Bias is any preference for choosing one concept (function) over another. Bias is a probability distribution over the set of functions expressed by the hypothesis space.*

Learning is the induction of a function which is (approximately) functionally equivalent to the underlying process. An unbiased learning system, by definition, is one where each function is equally likely to be produced. We state the definition given by [14]: "*Any basis for choosing one generalization over another, other than strict consistency with the observed training instances*".

6 Exploring the Nature of Hypothesis Spaces

6.1 Koza's Lens Effect

Koza [10] considers the role of representation. He examines the probability of generating a solution to the even-3-parity problem with three different types of representation; lookup tables, parse trees and Automatically Defined Functions. For lookup tables, there is the uniform chance of generating a correct lookup table of 1 in 256. Such a representation has no bias as each function is as likely as any other function to be generated. Given the function set {AND, OR, NAND, NOR}, he generates 10^7 trees at random but finds no solutions. However, if two, two argument Automatically Defined Functions are used, 35 solutions are generated. Koza calls this difference in the probability of generating a function due to the representation *"the lens effect"* (we call it representational bias), and talks about the problem environment being viewed through "*the lens of a given type of representation*". The difference between the parse tree results and the Automatically Defined Functions results is due entirely to the representation.

6.2 Langdon's Program Spaces

Langdon [12] generates the hypothesis spaces for 2, 3, and 4 input programs using the function set {NAND} and plots the proportion of programs that represent different functions. With the 2 input function plots, the functions are in lexical order, and he points out that this is not the most convenient ordering, so in other plots they are presented in order of decreasing frequency. From the plots for 3 and 4 inputs, it can be seen that more complex functions occur with less frequency than simpler functions. All Boolean functions of 3 inputs are considered when using base functions of {AND, OR, NAND, NOR} and {AND, OR, NAND, NOR, XOR}. As a general rule, we can see from these two plots (figures 5 and 6 in [12]) that more complex functions are represented less frequently. As it is impractical to analyze such large program spaces generated by 6 input programs, Langdon concentrates on just two functions, namely the always-on-6 function and the even-6-parity function. In this set of figures, [12] plots the error but again a correlation between the error and frequency can be observed.

Langdon [12] uses a polynomial of degree 6 for function regression with a function set $\{+, -, *, \%\}$ (where % is divide, protected against division by zero) and a terminal set consisting of the input x and random constants. Plots of the mean error against the proportion of times that error is observed are presented for increasing program length. Again, similar to the situation with Boolean program spaces, there is a correlation between error and the proportion of programs.

The artificial ant problem is a standard benchmark problem in the genetic programming literature. In some ways this is more complex than the previous two domains of logic and arithmetic functions as it includes side effects and iteration, that is it is Turing Complete. A plot of fitness against program length against proportion is presented. Again there is a correlation between the fitness and the proportion of program corresponding to that fitness similar to that seen with the logical functions and symbolic regression.

7 Justification of Occam's Razor

A preference for one function over another is expressed as $p(f_1) > p(f_2)$ where, the $p(f_1)$ is the probability of function f_1. We denote the complexity of a function

f as $c(f)$, and $c(f_1) < c(f_2)$ meaning, f_1 is less complex than f_2. We combine these to provide a statement of Occam's razor;

Theorem 1 (Occam's razor). $p(f_1) > p(f_2) \leftrightarrow c(f_1) < c(f_2)$

7.1 Occam's Razor, uniform sampling of program space

We begin by defining our notation. P is the hypothesis space (i.e. a set of programs). $|P|$ is the size of the space (i.e. the cardinality of the set of programs). F is the concept space (i.e. a set of functions represented by the programs in P). $|F|$ is the size of the space (i.e. the cardinality of the set of functions). If two programs p_i and p_j map to the same function (i.e. they are interpreted as the same function, $I(p_i) = f = I(p_j)$), they belong to the same equivalence class (i.e. $p_i \in [p_j] \leftrightarrow I(p_i) = I(p_j)$). The notation $[p_i]$ denotes the equivalence class which contains the program p_i (i.e. given $I(p_i) = I(p_j)$,then $[p_i] = [p_j]$). The size of an equivalence class $[p_i]$ is denoted by $|[p_i]|$.

We make two assumptions. The first assumption is that we uniformly sample the hypothesis space, and therefore, the probability of sampling a given program is $1/|P|$. This assumption can be relaxed somewhat and in discussed later in this section. The second assumption is that there are fewer hypotheses that represent complex functions: $|[p_1]| > |[p_2]| \leftrightarrow c(f_1) < c(f_2)$, where $I(p_1) = f_1$ and $I(p_2) = f_2$. Note that $|[p_1]|/|P| = p(I(p_1)) = p(f_1)$, that is $|[p_1]|/|P| = p(f_1)$ i.e. the probability of sampling a function is given by the ratio of the size of the equivalence class containing all the programs which are interpreted that function, divided by the size of the hypothesis space. Indeed, one can show that if there are many equivalent programs of the same length, then there must be a shorter equivalent program and a proof sketch of this assumption is given in [7].

Proof (Occam's Razor). We begin the proof by starting from the assumption;

$$|[p_1]| > |[p_2]| \leftrightarrow c(f_1) < c(f_2)$$

Dividing the left hand side by $|P|$,

$$|[p_1]|/|P| > |[p_2]|/|P| \leftrightarrow c(f_1) < c(f_2)$$

As $|[p_1]|/|P| = p(I(p_1)) = p(f_1)$, we can rewrite this as

$$p(f_1) > p(f_2) \leftrightarrow c(f_1) < c(f_2)$$

which is a statement of Occam's razor (theorem 1).

We take the probability distribution $p(f)$ to be the frequency with which the functions are represented in the hypothesis space (i.e. the size of the equivalence class). The assumption is $|[p_1]| > |[p_2]| \leftrightarrow c(f_1) < c(f_2)$, where $p(I(p_1)) = p(f_1)$, which we have not proved but is reasonable based on empirical work in [12]. We have not explicitly assumed more complex functions are less likely, but rather, the size of equivalence classes are larger if they contain programs which represent simple functions. For further empirical work supporting this see [16].

7.2 Occam's Razor, non-uniform sampling of program space

Above we assumed the hypothesis space was uniformly sampled. Now we make a much milder assumption, and theorem 1 is still true for a large number of distributions, not just a uniform distribution over the hypothesis space.

The probability of sampling some function is $p(f)$. Let the probability of sampling some function, given a certain hypothesis size be $p(f, s)$, and we know $p(f, s)$ is independent of size above some threshold [12]. Let the limiting distribution be $D(f)$. Thus, $p(f, s) = D(f)$, when $s > \theta$, where θ is the size threshold. S is the size of the largest hypotheses in the hypothesis space. Instead of uniformly sampling the search space, we weight hypotheses of size s with $w(s)$. The question becomes, what is the distribution of functions if we sample the hypothesis space with an arbitrary weighting over program size above the threshold.

Proof (Independence of Size). We begin the proof by stating the expression for the distribution of functions, weighted by $w(s)$

$$\textstyle\sum_{s>\theta}^{S} p(f, s).w(s)$$

As $p(f, s) = D(f)$, when $s > \theta$,

$$p(f, s). \textstyle\sum_{s>\theta}^{S} (w(s))$$

As $\sum_{s>\theta}^{S} (w(s)) = 1$, this reduces to $D(f)$. Therefore,

$$\textstyle\sum_{s>\theta}^{S} p(f, s).w(s) = D(f)$$

As the distribution over functions is independent of program size (above a certain program size), if we take any distribution over any program size (above the threshold), we obtain the same limiting probability distribution.

This second proof is vital to the full story. The first proof says, if we uniformly sample a program space, we encounter more simple functions than complex functions. The second proof says, if we sample a program space given programs of the same size equal weight (but programs of different size may have different weight), then we observe the same distribution of functions as if we had sampled the program space uniformly. In addition, there are issues if we attempt to sample over an infinite sized space; however we can take the limit as the size of the space increases (and this limit appears to exist [12]).

8 Summary

Occam's razor was a criticism of theories which became more complex without a corresponding increase in predictive power. i.e. the added complexity was redundant. As we are trying to prove a theorem which is about probabilities over functions, we are not in a position to make assumptions about probabilities over functions. such arguments can always be accused of circularity. This leaves us with the approach of looking at probabilities over programs.

From the empirical work of [12] we make two key observations (see Section 1). Firstly, simple functions can be expressed in more ways than complex function, and that there are more programs that map to simple functions than programs that map to complex functions. This suggests dividing the program space up into equivalence classes, where members of each class represent the same function. We make the assumption that the larger the equivalence class, the smaller the smallest program is in that set. The larger the equivalence class, the smaller the complexity of the function represented by programs in that class. As these equivalence classes are different sizes, uniformly sampling the program space will translate into a non-uniform sampling of the function space (i.e. a bias).

Secondly, as the limit of the of the programs in the hypothesis space increases, the proportion of programs that correspond to different functions does not change. The probability distribution tends towards some limiting distribution. What is more, this convergence is fast. Thus, in many machine learning paradigms, while a hard limit on the size of the hypothesis space is imposed, this does not affect the frequency with which functions are represented, as long as the size of the hypothesis space is larger than a certain threshold.

Occam's razor states a simpler function is more likely to generalize than a more complex function. While we agree; we argue that the underlying reason is that the simpler function is represented more frequently in the hypothesis space than more complex functions and this is the reason it should be chosen. We should prefer the function for reasons of probability rather than reasons of complexity, as it is ultimately probabilities we are interested in. We restate Occam's razor: *the most probable function is the one which belongs to the largest class of functionally equivalent hypotheses, consistent with the observed data.*

The most probable function to generalize the observed data is the one that is most *frequently* represented in the hypothesis space. By making observations, we can discard individual functions, but we cannot discard individual programs as we can only discard a whole equivalence class of functionally identical programs. There are two points to be made relating to this restatement of Occam's razor. Firstly, this is closer to the induction process. Secondly, if we take Occam's razor as "prefer the simpler", this gives us no indication of statistical confidence. Our approach, allows us to do this by considering the ratio of the sizes of the equivalence classes to which functions belong.

Occam's razor and NFL are mutually exclusive. In the former case there is a bias toward simpler functions, and in the latter case there is no bias. The NFL results should come as no surprise to the reader. Paraphrasing the theorems; if we *assume a uniform distribution over a set of all functions*, and after making some observations, then *there is still a uniform distribution over the remaining functions consistent with observation*, and hence we cannot prefer one function over another. Initially, if there is no bias, then finally there is no bias. With Occam's razor, we start with a monotonically decreasing distribution over a set of functions ordered by increasing complexity, and the act of observation dismisses some functions, then *there is still a monotonically decreasing distribution* over the remaining consistent functions. There are situations where NFL is applicable, and situations where Occam's razor is applicable.

I suspect that many people who regard NFL as the holy grail (as Hutter [7] so eloquently puts it), have not understood the assumptions, or the reasons for an alternative i.e. Occam's Razor. We would not attempt to predict next week's lottery numbers based only on the sequence of past lottery numbers as this is effectively white noise. We could, however, in principle, given the initial state of the lottery number generating machine, predict precisely next week's lottery numbers, as this initial information uniquely determines future states of the system. While Solomonoff's theory cannot be applied in practice, approximations can and are applied through out the machine learning literature. NFL, on the other hand, fails to have a single application (to the author's best knowledge). Indeed, nobody cares about the optimum of a white noise (incompressible) function [7]. Nor do we care about the optimum of very simple functions (e.g. constants). As Langdon [11] states; "most functions are constants and the remainder are mostly parsimonious". There is however a Goldilock zone of interesting functions which are sensible to apply machine learning algorithms to (i.e. not too simple, not too complex). Physicists are largely interesting in continuous functions [18], rather than incompressible functions. In addition NFL is not valid for continuous functions [1] or in a number of machine learning contexts (genetic programming, neural networks, hyper-heuristics) [17].

References

1. Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.
2. Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991.
3. Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Min. Knowl. Discov.*, 3(4):409–425, December 1999.
4. D. L. Dowe. MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness. *Philosophy of Statistics*, Handbook of the Philosophy of Science - (HPS Volume 7):901–982, 2011.
5. Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, November 2000.
6. Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. In *Machine Learning*, pages 63–91, 1993.
7. M Hutter. A complete theory of everything (will be subjective). *Algorithms*, 3(7):360–374, 2010.
8. Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2004. 300 pages, http://www.idsia.ch/~marcus/ai/uaibook.htm.
9. Michael J. Kearns and Umesh V. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, USA, 1994.
10. J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.
11. W. B. Langdon. Scaling of program functionality. *Genetic Programming and Evolvable Machines*, 10(1):5–36, March 2009.

12. William B. Langdon. Scaling of program fitness spaces. *Evolutionary Computation*, 7(4):399–428, 1999.
13. Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications (2nd ed.)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.
14. Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
15. Patrick M. Murphy and Michael J. Pazzani. Exploring the decision forest: An empirical investigation of occams razor in decision tree induction. *Journal of Artificial Intelligence Research*, pages 257–275, 1994.
16. S.L. Needham and D.L. Dowe. Message length as an effective ockham's razor in decision tree induction. In *Proc. 8th International Workshop on Artificial Intelligence and Statistics (AI+STATS 2001)*, pages 253–260, Key West, Florida, U.S.A., Jan. 2001, 2001.
17. Riccardo Poli, Mario Graff, and Nicholas Freitag McPhee. Free lunches for function and program induction. In *FOGA '09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, pages 183–194, Orlando, Florida, USA, 9-11 January 2009. ACM.
18. H. Rogers. *Theory of recursive functions and effective computability*. McGraw-Hill series in higher mathematics. MIT Press, 1987.
19. Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
20. C. Schaffer. A conservation law for generalization performance. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 259–265. Morgan Kaufmann, 1994.
21. Ray Solomonoff. Machine learning - past and future. In *AI@50*, pages 257–275, The Dartmouth Artificial Intelligence Conference, Dartmouth, N.H., 2006.
22. Ray J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964.
23. Ray J. Solomonoff. A formal theory of inductive inference. part ii. *Information and Control*, 7(2):224–254, 1964.
24. C. S. Wallace and D. L. Dowe. Minimum message length and kolmogorov complexity. *Computer Journal*, 42:270–283, 1999.
25. Webb. Generality is more significant than complexity: Toward an alternative to occam's razor. In *AJCAI: Australian Joint Conference on Artificial Intelligence*, 1994.
26. David H. Wolpert and William G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM, 1995.
27. David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.
28. John R. Woodward. Complexity and cartesian genetic programming. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 260–269, Budapest, Hungary, 10 - 12 April 2006. Springer.
29. John R. Woodward. Invariance of function complexity under primitive recursive functions. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 310–319, Budapest, Hungary, 10 - 12 April 2006. Springer.