Free Lunch for Optimisation under the Universal Distribution

Tom Everitt[1], Tor Lattimore[2], and Marcus Hutter[3]

[1]Stockholm University, Sweden. Email: everitt@math.su.se
[2]University of Alberta, Edmonton, Canada
[3]Australian National University, Canberra, Australia

Abstract

Function optimisation is a major challenge in computer science. The No Free Lunch theorems state that if all functions with the same histogram are assumed to be equally probable then no algorithm outperforms any other in expectation. We argue against the uniform assumption and suggest a universal prior exists for which there is a free lunch, but where no particular class of functions is favoured over another. We also prove upper and lower bounds on the size of the free lunch.

Keywords: *No Free Lunch; Black-box Optimisation; Universal Distribution; Solomonoff induction; Kolmogorov complexity*

1 Introduction

Finite black-box optimisation is the problem of finding an optimal value (usually the maximum or minimum) of a target function $f\colon X \to Y$ where X and Y are finite. A wide range of tasks may be formulated in this setting. For instance, drug-design may be viewed as the task of finding a mix of chemicals that maximises recovery chances. Since experimentation is expensive it is crucial that the best drug be found as soon as possible.

It is desirable to find optimisation algorithms that perform well on a wide variety of target functions, as this minimises the need for fine-tuning the algorithm to the problem. Indeed, several such algorithms exist and are regularly employed in practice; examples include hill-climbing and simulated annealing, as well as genetic algorithms. However, the theoretical understanding of the conditions permitting such "universal" algorithms remains limited [CO01, Str03, WR06, JC11]. To approach this problem, we derive bounds for expected optimisation-performance under assumptions justified in all (or virtually all) optimisation settings.

The original No Free Lunch (NFL) theorems state that when the global performance of an optimisation algorithm is measured by taking a uniform average of its performance over all functions from X to Y, then no algorithm is better than random (assuming no point is sampled more than once) [WM97]. The uniform assumption is justified by assuming the absence of prior knowledge and the results are often used to claim that no optimisation algorithm can be universal.

There is, however, another viewpoint. If we assume that the function $f\colon X \to Y$ to be optimised is generated by some (unknown) computer program, then taking a uniform prior over programs is arguably more natural. This is a reasonable assumption based on the commonly held view that the universe is likely to be (stochastically) computable [Fre92, Wol02, Hut12]. The distribution on functions induced by this approach is the famous universal lower-semicomputable semi-distribution[1] developed by Solomonoff and others [LV08]. The universal distribution satisfies many nice properties, both theoretical and philosophical. It is a natural choice when formalising Occam's razor in combination with Epicurus' principle of multiple explanations since it favours simplicity over complexity without disregarding the possibility that the truth is complex [Hut05, RH11]. The universal distribution also exhibits a range of other desirable properties, discussed further in Section 3.

If performance is measured in expectation with respect to the universal distribution, then the no free lunch theorems can no longer be applied. Indeed, under some highly technical conditions Streeter [Str03] showed that there is a free lunch for optimisation under Solomonoff's universal distribution. Tightly related to the universal distribution is Kolmogorov complexity: Borenstein and Poli [BP06] discuss Kol-

[1] The use of lower semicomputable semi-distributions rather than regular computable distributions is technical only and may be ignored by the reader unfamiliar with algorithmic information theory.

Published as: pp. 167–174, in Proceedings of 2014 IEEE Congress on Evolutionary Computation (CEC), July 6-11, 2014, Beijing, China. DOI: 978-1-4799-1488-3/14/$31.00 ©2014 IEEE

mogorov complexity and optimisation, and also give a good account of previous research in this area (see also [McG06]). Several authors report on Kolmogorov complexity not being perfectly related to searchability [SVW01, DJW02, BP06], but except for [Str03], implications for search performance under the universal distribution have not been investigated. The relation between the universal distribution and the NFL theorems for supervised learning has been studied by Lattimore and Hutter [LH11]. In sequence prediction and reinforcement learning, the universal distribution has been extensively researched [Hut05].

We first improve on [Str03] by presenting the first easily interpretable theorem that there is a free lunch if performance is measured in expectation with respect to Solomonoff's universal distribution rather than the uniform distribution originally used by Wolpert and Macready (Section 6). Unfortunately the size of the free lunch turns out to be somewhat limited. Under only weak assumptions we show that no computable algorithm can perform much better than random, even when performance is averaged with respect to the universal distribution (Section 7). This result is then extended to arbitrary (possibly non-computable) optimisation algorithms for a commonly used performance measure.

2 Preliminaries

A *(finite binary) string* is a finite sequence $x = b_1 b_2 \cdots b_n$ with $b_i \in \mathbb{B} = \{0, 1\}$ and length $\ell(x) = n$. The set of all finite binary strings is denoted by $\mathbb{B}^*$. Strings may be concatenated in the obvious way. Power notation is used to represent multiple concatenations: for example, $01^4 0 = 011110$.

A *problem context* is a pair X, Y of finite subsets of $\mathbb{B}^*$, both containing at least 0 and 1 (to avoid degenerate cases). In a problem context X, Y, the *search space* is X and the *range* is Y. We let $\mathcal{X}$ and $\mathcal{Y}$ be the sets of all search spaces and all ranges respectively.

Definition 1. An *optimisation problem* is a collection $\mathrm{P} = \{ P_{XY} \}$, where P_{XY} is a measure over the finite set $Y^X = \{ f \colon X \to Y \}$ of functions from X to Y.

A *search trace* T_n *on* X, Y is an ordered n-tuple $\langle (x_1, y_1), \ldots, (x_n, y_n) \rangle \in (X \times Y)^n$, representing a search history. The empty search trace will be denoted $\langle\rangle$. Let $\mathcal{T}_n(X, Y)$ be the set of all search traces of length n, and let $\mathcal{T}(X, Y) \coloneqq \bigcup_{i=0}^{|X|} \mathcal{T}_i(X, Y)$ be the set of traces of any length. Further, let $\mathcal{T} \coloneqq \bigcup_{X,Y \in \mathcal{X} \times \mathcal{Y}} \mathcal{T}(X, Y)$ be the set of search traces on any context. If $T_n = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle$, then $T_n^x \coloneqq \langle x_1, \ldots, x_n \rangle$ and $T_n^y \coloneqq \langle y_1, \ldots, y_n \rangle$.

An *optimiser* is a function $a \colon \mathcal{X} \times \mathcal{Y} \times \mathcal{T} \to \mathcal{X}$ where $a(X, Y, T) \in X - T^x$ for all (X, Y, T). The optimiser selects new, unvisited search points in the search space based on previously seen data and the problem context. That the optimiser is only permitted to sample unvisited points is standard in the literature, and non-restrictive in the noise-free setting considered in this paper.

The setup is this: A problem context X, Y is fixed, and a target function $f \colon X \to Y$ is sampled from Y^X according to the problem distribution P_{XY}. The optimiser is initialised with the empty search trace and the problem context, and outputs a search point $x_1 \in X$ by $a(X, Y, \langle\rangle) = x_1$. The search trace becomes $\langle x_1, f(x_1) \rangle$. The new search trace is fed to the optimiser, which produces a new search point x_2 via $a(X, Y, \langle (x_1, f(x_1)) \rangle)$, and so on. Observe that the search trace is a function of the optimiser, the problem context and the sampled function f. We write $T_{XY}(a, f)$ for the "full" trace of length $|X|$ that a generates on f and X, Y; when X, Y is clear from the context, $T(a, f)$ will suffice. The Y-components $T_{XY}^y(a, f)$ will be called the *result vector* of a on f and X, Y. We will also use R to denote result vectors. Let $\mathcal{R}(X, Y)$ be the set of all result vectors on X,Y, and let $\mathcal{R} = \bigcup_{X,Y \in \mathcal{X} \times \mathcal{Y}} \mathcal{R}(X, Y)$.

The performance of an optimiser a on a problem P is measured in terms of the result vectors it produces. A function $M \colon \mathcal{X} \times \mathcal{Y} \times \mathcal{R} \to [0, \infty)$ defines a *performance measure* by the P-expected value of M for a on each problem context X,Y:

$$M_{XY}^{\mathrm{P}}(a) \coloneqq \sum_{f \in Y^X} \mathrm{P}_{XY}(f) M_{XY}(T^y(a, f)) \qquad (1)$$

We use $[\![s]\!]$ for the Iverson bracket that is 1 when s is true, and 0 otherwise. For any list R, $R[i]$ extracts its ith element.

3 The universal distribution

We now give a short introduction to Kolmogorov complexity and the universal distribution. Detailed references are [LV08] and [Cal02].

Prefix codes are central elements in algorithmic information theory. A *prefix code* is a set of *code words* (formally, strings) where no code word is a prefix of another. This makes prefix codes *uniquely decodable*: in a sequence of appended code words it is possible to tell where one code word ends another begins. Kraft's inequality gives a lower bound on the length of the code words in a prefix code [LV08, p. 76]. Definition 2 gives some commonly used prefix encodings of strings, numbers, lists and functions.

Definition 2 (String encodings)**.** Let x be a binary string, n a natural number and $Z = z_1, \dots, z_n$ a list of strings. Then $\bar{x} := 1^{\ell(x)}0x$, $\bar{n} := 1^n 0$ and $\bar{Z} := \bar{n}\bar{z}_1 \cdots \bar{z}_n$ defines prefix codes for x, n and Z. The code for lists may be applied recursively to lists of lists. Target functions $f\colon X \to Y$ are encoded by lists $f(x_1), \dots, f(x_n)$ where $x_1, \dots, x_n$ are the elements of X in order.

For technical reasons, regular Turing machines are not suitable for defining the universal distribution, so prefix machines are often used instead.

Definition 3 (Prefix Machines)**.** A *prefix machine* V is a Turing machine with one unidirectional input tape, one unidirectional output tape, and some bidirectional work tapes. Input tapes are read only, output tapes are write only. All tapes are binary and work tapes are initialised with zeros. We say V halts with output x on input p given s and write $V(p|s) = x$, if $\bar{s}p$ is to the left of the input head and x is to the left of the output head when V halts. For any $s \in \mathbb{B}^*$, the inputs on which $V(\cdot|s)$ halts form a prefix code. Also, just as for regular Turing machines, there are *universal prefix machines* that can simulate any other prefix machine.

Definition 4 (Prefix Complexity)**.** Let $x, y \in \mathbb{B}^*$ be finite binary strings and U a universal prefix machine, then the *Kolmogorov complexity of x conditioned on y* is the length of the shortest program that given y outputs x.

$$K_U(x|y) := \min\{\, \ell(p) : U(p|y) = x \,\} \qquad (2)$$

A simple but fundamental theorem is that K_U depends on U only up to constant factors, so from now on, as is usual in algorithmic information theory, we fix an arbitrary universal prefix machine as a *reference machine* and simply write $K(x)$ for $K_U(x)$.

Definition 5 (Function complexity)**.** Let $f\colon X \to Y$, then the *complexity of f* is $K(f|X, Y)$, with f and X, Y encoded by strings according to Definition 2.

The Martin-Löf–Chaitin Thesis states that randomness may be defined as incompressibility [GTW+11, p. 705]. A target function is *incompressible* or *random* if $K(f|X, Y) \geq |X| \log |Y|$. A classical result in algorithmic information theory shows that almost all functions are (nearly) random. Thus, the uniform distribution puts the majority of its weight on random functions, which is one explanation for why it is hard to optimise under the uniform distribution. In contrast, the universal distribution puts more weight on "simple", non-random functions:

Definition 6 (Universal distribution)**.** For each context X, Y, the *universal distribution* is defined as

$$\mathbf{m}_{XY}(f) := c_{\mathbf{m}_{XY}} \cdot 2^{-K(f|X,Y)}, \qquad (3)$$

where $c_{\mathbf{m}_{XY}} = 1/\sum_{f\colon X \to Y} 2^{-K(f|X,Y)}$ is just a normalising constant. In the literature, unnormalised versions of $\mathbf{m}$ are often considered. Although $c_{\mathbf{m}_{XY}}$ may fluctuate with X, Y, there is a constant $c_{\mathbf{m}}$ depending only on the reference machine such that $1 \leq c_{\mathbf{m}_{XY}} \leq c_{\mathbf{m}}$ for all X, Y. Note that $\mathbf{m}$ is an optimisation problem, since $\mathbf{m}_{XY}$ is a distribution over Y^X for each X, Y.

Somewhat surprisingly, there is an equivalent definition of $\mathbf{m}$ as the distribution obtained by feeding random coin-flips into a universal prefix machine with access to X, Y.

$$\mathbf{m}_{XY}(f) \approx \sum_{p \in \mathbb{B}^*} 2^{-\ell(p)} [\![V(p|X, Y) = f]\!]. \qquad (4)$$

The approximation holds up to irrelevant multiplicative factors, so (4) is often used in place of (3) as the definition of the universal distribution.[2] Feeding a universal prefix-machine random coin-flips is a natural formalisation of the uniform prior over computer programs advocated in the introduction. Thus $\mathbf{m}$ may be justified as a *subjective prior* for the assumption that the target function has been computably generated.[3]

The bias away from randomness also aligns with our intuition of how functions "ought to be optimised". If the first hundred observations are predicted by a simple polynomial, then common sense (and Occam's razor) suggests that the best prediction of unseen points is that they follow the polynomial. In general, the "simplest" structure perceivable in the data should be the most likely extrapolation. The universal distribution is consistent with this intuition. A detailed discussion of the philosophical justification for the universal prior can be found in [RH11].

To summarise, we have argued for the universal distribution on the following grounds:

- (Weak assumptions): If the target function is generated by a computer program, then a uniform prior over computer programs is justified. Formalised as in (4), this yields the universal distribution.

[2] Even the definition given in (3) depends on the choice of reference machine up to multiplicative factors.

[3] There are also "objective" grounds to prefer $\mathbf{m}$ as a prior, including *regrouping invariance* [Hut07] and *dominance* [LV08]. Neither hold for the uniform distribution.

- (Downweighs randomness): A uniform prior over target functions puts the (vast) majority of the weight on (nearly) random functions. The universal distribution concentrates on structured functions, without favouring any particular class of functions.
- (Aligns with Occam): Intuition and Occam's razor suggests that the best extrapolation is the continuation of the "simplest" pattern observable in data, which corresponds well with the relative probabilities of the universal distribution.

Next we will present some background on the NFL theorems and introduce our performance measure M_{ot}, before taking a closer look at optimisation under $\mathbf{m}$.

4 No Free Lunch

The NFL theorems provide important insights into the possibility of universal optimisation. They show that for certain distributions P_{XY} all optimisers perform identically with respect to some (or all) performance measure(s). This is often phrased as "there is no free lunch available for P_{XY}". For example, if NFL holds for a performance measure depending on how many function evaluations are required to find the maximum, then this implies that in expectation a hill-climbing optimiser will find the maximum as slowly as a hill-descending optimiser.

Definition 7 (Performance measure-NFL)**.** *NFL* holds for a distribution P_{XY} and a performance measure M if $M^{\mathrm{P}}_{XY}(a) = M^{\mathrm{P}}_{XY}(b)$ for all optimisers a and b. If NFL holds for *all* performance measures M, then NFL simply holds for P_{XY}. If NFL does not hold for P_{XY} (and M), then we say that there is *free lunch* for P_{XY} (and M).

The stronger statement that NFL holds for all performance measures may equivalently be defined in terms of result vectors. The proof of the equivalence is a straightforward application of the definitions.

Lemma 8 (Result vector-NFL)**.** *Let $\mathrm{P}_{XYa}(R)$ be the probability $\sum_{f\in Y^X} \mathrm{P}_{XY}(f)[\![T^y(a,f) = R]\!]$ that an optimiser a generates the result vector R, then NFL holds for P_{XY} if and only if $\mathrm{P}_{XYa}(R) = \mathrm{P}_{XYb}(R)$ for every pair of optimisers a and b and every result vector $R \in Y^{|X|}$.*

4.1 The NFL theorems

Igel and Toussaint [IT04] showed that the precise condition for NFL is *block uniformity* of P_{XY}.

Definition 9 (Block uniformity)**.** A *histogram* for a function f is a function $h_f\colon Y \to \mathbb{N}$ defined as $h_f(y) = |f^{-1}(y)|$, indicating how many x's map to every y. The subset of all functions $X \to Y$ with histogram h is called the *base class* of h, and is denoted B_h. The distribution P_{XY} is *block uniform* if for every h it holds that $f, g \in B_h \implies \mathrm{P}_{XY}(f) = \mathrm{P}_{XY}(g)$.

Theorem 10 (Non-uniform NFL [IT04])**.** *NFL holds for P_{XY} if and only if P_{XY} is block uniform.*

The original NFL theorem by Wolpert and Macready [WM97] showed that NFL holds when P_{XY} is uniform on Y^X. As uniform distributions are a special case of block uniform distributions, Wolpert and Macready's result follows from Igel and Toussaint's.

Another special case is the NFL theorem for uniform optimisation problems over function classes closed under permutation (c.u.p.) by Schumacher et al. [SVW01]. A *permutation* is a bijective function $\sigma\colon X \to X$ that permutes functions via $(\sigma f)(x) = f(\sigma^{-1}(x))$. A class $F \subseteq Y^X$ is *c.u.p.* if $f \in F \implies \sigma f \in F$ for all permutations σ. The uniform distribution u_F over F is defined by $u_F(f) := 1/|F|$ if $f \in F$ and 0 otherwise.

Theorem 11 (NFL for c.u.p. classes [SVW01])**.** *If P_{XY} is the uniform distribution over a class $F \subseteq Y^X$, then NFL holds for P_{XY} if and only if F is c.u.p.*

A simple consequence of the NFL theorems is that all optimisers produce the same result vectors. We state this as a lemma for future reference.

Lemma 12 ([SVW01])**.** *The set of result vectors $\{\, R \in \mathcal{R}(X,Y) : T^y(a,f) = R \text{ for some } f \in Y^X \,\}$ ever produced by an optimiser a on X, Y is the same for all optimisers.*

The NFL theorems have also been investigated in infinite and continuous domains. Depending on the generalisation, free lunches may or may not emerge in those settings [AT07, RVW09].

5 Performance measures

So far we have only considered problems for which either NFL holds for all performance measures, or for which a free lunch is available for some performance measures. Often, however, we are interested in performing well under a fixed performance measure of interest. One natural choice of performance measure is *optimisation time*, which in this context means the number of function evaluations required to find the maximum.[4]

Definition 13. The *optimisation time performance measure* M_{ot} is defined as

$$M_{\mathrm{ot},XY}(R) := \min_i \left(R[i] = \max Y\right),$$
$$M^{\mathrm{P}}_{\mathrm{ot},XY}(a) := \sum_{f\in Y^X} \mathrm{P}_{XY}(f) M_{\mathrm{ot},XY}(T^y(a,f))$$

for result vectors and optimisers, respectively. Under M_{ot} a low score is better than a high score.

A variety of performance measures have been considered in the literature. Some use properties of the k first function evaluations, for example the number of values exceeding a certain threshold [CO01, WR06, JC11], or the probability that some seen value exceed the threshold [WM97]. Griffiths and Orponnen [GO05] use a performance measure $M_{\max}$ depending on the size of the greatest value of the first k observations. Others, such as [BP06, Jan13], use M_{ot}. The main reason we prefer M_{ot} to the other alternatives is that it is better suited for the asymptotic results we will aim for.

Results about particular performance measures often have greater practical interest than their arbitrary-measure counterparts. In addition, particular performance measures may also have theoretical interest. Under particular performance measures, NFL may hold for classes that are not c.u.p. [GO05]. This does not contradict Theorem 11, which only claims that for every non-c.u.p. class, there is *some* performance measure permitting a free lunch. Indeed, it is unsurprising that NFL will apply to wider ranges of function classes when a fixed performance measure is used. The conditions for NFL under Griffiths and Orponnen's performance measure $M_{\max}$ turn out to be significantly more intricate compared to the standard NFL case [GO05].

Another difference is found in the "cleverness" required to exploit a free lunch. Optimisers that choose the next point to probe irrespective of previous observations are called *non-adaptive*; such optimisers can only exhibit a limited amount of sophistication. Proposition 14 shows that when a free lunch is available and arbitrary measures are allowed, then there is free lunch for a non-adaptive optimiser. In contrast, under particular measures such as M_{ot} and $M_{\max}$, adaptive optimisers may differ in performance while all non-adaptive optimisers perform the same. In this sense, "smarter" algorithms may be required for exploiting a free lunch when using a particular performance measure, compared to when arbitrary performance measures are permitted.

Proposition 14. *If NFL does not hold for a distribution* P_{XY}*, then there is free lunch for a non-adaptive optimiser under some performance measure.*

Proof. Since NFL does not hold for P_{XY} we have by Theorem 10 that P_{XY} is not block uniform. Hence there are two functions f and σf in the same base class B_h such that $\mathrm{P}_{XY}(f) > \mathrm{P}_{XY}(\sigma f)$, where σ is a permutation on X. Let e and e_σ be non-adaptive optimisers, with e searching $X = \{x_1, \dots, x_n\}$ in order, and e_σ searching X in the order of $\sigma X = \{\sigma(x_1), \dots, \sigma(x_n)\}$. Now e generates the result vector $R_f = \langle f(x_1), \dots, f(x_n)\rangle$ exactly when f is the true function, and e_σ generates R_f exactly when σf is the true function. An immediate consequence is that $\mathrm{P}_{XYe}(R_f) = \mathrm{P}_{XY}(f) > \mathrm{P}_{XY}(\sigma f) = \mathrm{P}_{XYe_\sigma}(R_f)$. That is, the non-adaptive algorithms e and e_σ generate R_f with different probability, which means that there is free lunch for some non-adaptive optimiser under some performance measure by Lemma 8. □

In conclusion, specific performance measures can be considered for both practical and theoretical reasons. They are more practically relevant in the sense that they measure aspects we care about in practice (such as how long it takes to find a maximum). But they also have theoretical interest, as they expose aspects that are invisible from an arbitrary-measure perspective.

6 Universal Free Lunch

We now turn to the question of whether or not a free lunch is available under $\mathbf{m}$, which we will answer in the affirmative for both arbitrary performance measures and M_{ot}.

The universal distribution solves the induction problem for sequence prediction [Hut05, RH11]. Black-box optimisation also include an induction problem in the extrapolation of target-function behaviour from the points already evaluated. Although successful inference of the target-function behaviour may not be strictly necessary, it will typically enable better choices of future search points.

There are several important differences between sequence prediction and optimisation. First, optimisation is an *active* setting: the choices of the optimiser affect both the learning outcome and the reward. This entails an exploration/exploitation

[4] In black-box optimisation in general, and evolutionary algorithms in particular, the evaluation of the target function typically constitutes the main expense of computation time. This is the motivation behind the name *optimisation time* for the number of target function evaluations.

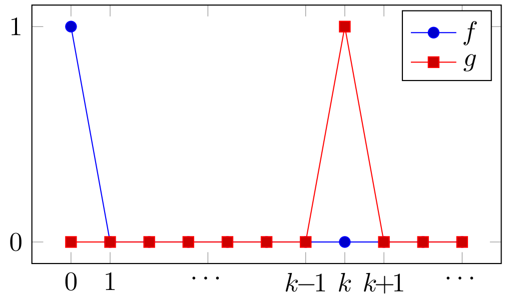

Figure 1: The function f has complexity bounded by a constant c_f independent of X and Y. In contrast, the complexity of g grows logarithmically with $|X|$. See the proof of Theorem 15 for details.

tradeoff in the choice between potentially informative points and points likely to mean high performance (e.g. points likely to be a maximum). Further, optimisation is a finite setting, which yields less time to exploit a good model (compared to sequence prediction where infinite sequences are considered). There are also major differences in the hypothesis classes and in how performance is measured.

Section 7 presents a number of results bounding the amount of free lunch under $\mathbf{m}$. Perhaps surprisingly, only a small amount of free lunch is available under the universal distribution.

6.1 Free lunch under arbitrary performance measures

Streeter [Str03] showed that there is free lunch for $\mathbf{m}$ under certain technical conditions. We prove a similar result, but with more interpretable conditions (in terms of the size of X, only). We also use a different proof than Streeter.

Theorem 15 (Universal free lunch)**.** *There is free lunch for the problem* $\mathbf{m}$ *for all problem contexts with sufficiently large search space (the required size depending on the reference machine only).*

Proof. It will be shown that $\mathbf{m}_{XY}$ is not block uniform for problem contexts with sufficiently large X, which by Theorem 10 implies that NFL does not hold.

Pick a problem context X,Y. Consider two functions f and g in the base class $B_h \subseteq Y^X$ of functions with one value 1 and the rest of the values 0. Let f be 1 at x_1 and let g be 1 at some point x_k chosen so that $K(g\,|\,X,Y) \geq \log_2|X| - 1$ (see Fig. 1). To see that such a g exists, note that there are $|X|$ different functions in B_h. As the halting programs for the reference machine constitute a prefix code, there can be at most $|X|/2$ halting programs of length $\leq \log_2|X| - 1$ by Kraft's inequality. Thus at least one of the B_h-functions must have a shortest program longer than $\log_2|X| - 1$, and therefore complexity $K(g\,|\,X,Y) \geq \log_2|X| - 1$. Meanwhile, $K(f\,|\,X,Y) \leq c_f$ for some constant c_f independent of the problem context. So for search spaces with $\log_2(|X|) - 1 > c_f$, this means that f will have lower complexity than g, and thus that $\mathbf{m}_{XY}$ will assign different probabilities to f and g. As f and g are elements of the same base class, this shows that $\mathbf{m}$ is not block uniform for search spaces greater than 2^{c_f+1}. By Theorem 10, this implies a free lunch for $\mathbf{m}$ under some performance measure. □

Indeed, $\mathbf{m}$ is not even close to block uniform for large search spaces in the sense that the functions of type f and g will receive substantially different weights. However, this does not necessarily imply a big free lunch, as we shall see in Section 7.

6.2 Free lunch under M_{ot}

As has been discussed, in practice we often care about a particular performance measure such as M_{ot}.

Theorem 16. *There is free lunch for the problem* $\mathbf{m}$ *under the performance measure* M_{ot} *for all problem contexts with sufficiently large search space (the required size depending on the reference machine only).*

The proof is similar to Theorem 15, but more work is required to ensure a complexity difference between two potential maximums, rather than between two specific functions. A full proof is included in the Appendix.

7 Upper Bounds

Theorems 15 and 16 show that there is free lunch under the universal distribution. This section will bound the amount of free lunch available, and show that it is only possible to outperform random search by a constant factor. First we show that the performance of computable optimisers deteriorates linearly with the worst-case scenario and the size of the search space. This result applies to decidable performance measures in general, and has a concrete interpretation for M_{ot}, where it implies that as the size of the domain is increased, a non-zero fraction of the domain must be probed before a maximum is found in expectation. This does not contradict the free lunches above, as the required fraction may differ between optimisers.

We also consider possible ways to circumvent the negative result described above by means of incomputable search procedures. A further negative result for M_{ot} is obtained: It does not appear possible to find the maximum with only $o(|X|)$ target function evaluations. That is, the expected number of probes required to find the maximum grows linearly with the size of the search space, but again, the proportion may differ substantially between optimisers.

7.1 Computable optimisers

To bound the amount of free lunch available for computable optimisers, we will adapt a proof-technique for showing that average-case complexity is equal to the worst-case complexity under the universal distribution [LV08, Section 4.4]. Although no formal theorem relies on it, we will think of greater M-values as *worse* performance.

Lemma 17. *A function $f_{\text{bad}}\colon X \to Y$ is* maximally bad *for an optimiser a on the problem context X,Y with respect to a performance measure M if $M_{XY}(T^y(a, f_{\text{bad}})) = \max_{R\in\mathcal{R}(X,Y)} M_{XY}(R)$. There always exists a maximally bad function $f_{\text{bad}}\colon X \to Y$ for a with respect to M, regardless of the performance measure M, the optimiser a and the problem context X,Y.*

Proof. By Lemma 12, all optimisers produce the same result vectors, so it suffices to show that *some* optimiser has a maximally bad function. The non-adaptive optimiser e that searches X in order has a maximally bad function. To see this, let R_{bad} be a maximally bad result vector on X,Y, and let f be the function satisfying $f(x_i) = R_{\text{bad}}[i]$ for all $x_i \in X$. Then e produces R_{bad} on f. □

A performance measure M for which there is an algorithm deciding whether $M_{XY}(R_1) < M_{XY}(R_2)$ for every X,Y and every pair of result vectors $R_1, R_2 \in \mathcal{R}(X,Y)$ is *decidable.* For any decidable performance measure M, it is possible to create a procedure $\textsc{FindWorst}(a, X, Y)$ that given a computable optimiser a (specified by some binary string) and a context X, Y, returns a maximally bad function $f_{\text{bad}}\colon X \to Y$ for a. $\textsc{FindWorst}$ is a computable operation since a is computable and M is decidable: $\textsc{FindWorst}$ need only simulate a on all possible functions in Y^X, and output one that yields a worst result vector. This shows that for all decidable performance measures, all computable optimisers a and all contexts X, Y, there is a maximally bad function $f_{\text{bad}}\colon X \to Y$ for a with complexity

$$K(f_{\text{bad}}\,|\,X, Y) \leq \ell(\textsc{FindWorst}) + \ell(a) + c \ , \quad (5)$$

where the c term depends only on the reference machine, and absorbs the cost for initialising $\textsc{FindWorst}$ with a, X and Y. Pivotally, the bound is independent of X, Y. This is the central observation behind the following theorem, which shows that expected performance always deteriorates linearly with the worst-case scenario. The theorem's prime relevance is for performance measures whose worst-case value grows with X.

Theorem 18 (Almost NFL for $\mathbf{m}$)**.** *For every decidable performance measure M and every computable optimiser a there exists a constant $c_a > 0$ such that for all X, Y*

$$M^{\mathbf{m}}_{XY}(a) \geq c_a \max_{R\in\mathcal{R}(X,Y)} M_{XY}(R) \ .$$

Proof. Let X,Y be a problem context and f_{bad} be the output of $\textsc{FindWorst}(a, X, Y)$, then

$$\begin{aligned} M^{\mathbf{m}}_{XY}(a) &= \sum_{f\in Y^X} \mathbf{m}_{XY}(f) M_{XY}(T^y(Y, a, f)) \\ &\geq c_{\mathbf{m}_{XY}} 2^{-K(f_{\text{bad}}|X,Y)} M_{XY}(T^y(a, f_{\text{bad}})) \ , \end{aligned}$$

where we have first used the definition (1) of performance measures, and then that the sum of non-negatives is greater than all of its terms. But $c_{\mathbf{m}_{XY}} 2^{-K(f_{\text{bad}}|X,Y)} \geq c_a$ for some $c_a > 0$ independent of X, Y due to $c_{\mathbf{m}_{XY}} \geq 1$ and the complexity bound (5). And $T^y(a, f_{\text{bad}})$ was a worst result vector by the construction of $\textsc{FindWorst}$. Combined, this gives the bound $M^{\mathbf{m}}_{XY}(a) \geq c_a \cdot \max_{R\in\mathcal{R}(X,Y)} M_{XY}(R)$. □

This theorem shows that for every performance measure M, there is only a constant amount of free lunch available in an asymptotic sense. It has no impact on performance measures whose worst-case value does not grow unboundedly with either X or Y. However, the "semi-assumption" of higher values being worse is not necessary: If the converse is the case and high values are better, then the proposition shows that all optimisers will do well. Indeed, this is also an NFL result, as it implies that random search (and even optimisers designed to do poorly!) will perform well.

Applied to the performance measure M_{ot}, Theorem 18 has a fairly concrete interpretation: For any computable optimiser a, the expected number of evaluations to find the maximum grows linearly with $|X|$. Corollary 19 follows immediately from Theorem 18 and the observation that for any context X, Y, the worst-case scenario is to find a maximum only at the very last probe; that is, $\max_{R\in\mathcal{R}(X,Y)} M_{\text{ot},XY}(a) = |X|$.

Corollary 19. *For every computable optimiser a there exists a constant $c_a > 0$ such that $M^{\mathbf{m}}_{\text{ot},XY}(a) \geq c_a \cdot |X|$ for all optimisers a and all problem contexts X,Y.*

The implications of this result should not be overstated. The constant c_a may be very small; for example, if the description of the optimiser a is 100 bits long, then c_a becomes of the order 2^{-100}. The fact that the expected number of probes is forced to grow linearly with such a constant is mainly of theoretical importance. Nonetheless, the result does illustrate a fundamental hardness of optimisation, and shows that the universal distribution does not provide enough bias for efficient (sublinear) maximum finding.

7.2 Needle-in-a-haystack functions

A problematic class of functions is the class of so-called needle-in-a-haystack (NIAH) functions. We will use them to generalise Corollary 19 to incomputable optimisers. A *NIAH-function* is a target function that is 0 in all points except one where it equals 1. The exception point is called the *needle.* It should be intuitively clear that it is hard to find the maximum of a NIAH-function. Probing a NIAH-function, the output will generally just turn out to be 0 and provide no clues to where the needle might be.

Formally, for any X, Y let NIAH_{XY} be the class of NIAH functions on X, Y and let u_{NIAH} be the *uniform NIAH problem* defined as $u_{\mathrm{NIAH},XY}(f) := 1/|\mathrm{NIAH}_{XY}|$ if $f \in \mathrm{NIAH}_{XY}$ and 0 otherwise. The function class NIAH_{XY} is c.u.p. for any X, Y, so NFL holds for $u_{\mathrm{NIAH},XY}$ by Theorem 11. The expected performance (of any optimiser) on the uniform NIAH-problem can be calculated from a general result of Igel and Toussaint [IT03]. They show that for any c.u.p. problem u_F where F only contains functions with exactly m maxima, the expected number of probes to find a maximum is $(|X|+1)/(m+1)$. The NIAH-functions have exactly one maximum, which gives the following lemma.

Lemma 20. *Under $u_{\mathrm{NIAH},XY}$, the expected optimisation time is $M^{u_{\mathrm{NIAH}}}_{\mathrm{ot},XY}(a) = (|X|+1)/2$ for any optimiser a.*

One feature that makes the NIAH-class more problematic than other c.u.p. function classes is that the NIAH-functions all have fairly low complexity (as remarked by [SVW01, BP06]). The NIAH-functions have low complexity, since to encode a NIAH-function one only needs to encode that it is NIAH (which takes a constant number of bits) and the position of the needle (which requires at most $\log_2|X|$ bits). A NIAH-function thus has complexity of order $O(\log_2|X|)$; in comparison, a random function has exponentially greater complexity (above $|X|\log_2|Y|$).

The NIAH-measure is computable. This is intuitively obvious, and easily verified against the formal definitions of computable functions. Definitions of real-valued computable functions can be found in [LV08]. It is well-known that $\mathbf{m}$ *dominates* any computable measure in the following sense.

Lemma 21 ($\mathbf{m}$ dominates u_{NIAH})**.** *There is a constant $c_{\mathrm{NIAH}} > 0$ such that for all X, Y and all functions $f\colon X \to Y$, it holds that $\mathbf{m}_{XY}(f) \geq c_{\mathrm{NIAH}} \cdot u_{\mathrm{NIAH},XY}(f)$.*

7.3 Incomputable optimisers

Theorem 18 and Corollary 19 were proven for computable optimisers. We now show that even incomputable optimisers suffer a linearly growing loss in $|X|$ when the performance measure is M_{ot}. Incomputable search procedures may seem like remote objects of concern, but for example the (Bayes-)optimal procedure for $\mathbf{m}$ is incomputable due to the incomputability of $\mathbf{m}$. Therefore, incomputable procedures do at least have theoretical interest.

The following theorem generalises Corollary 19 to incomputable search procedures, showing that they also must search a linearly growing portion of X to find the maximum. The theorem does not generalise to arbitrary performance measures however, so the analogous generalisation of Theorem 18 may not be true.

Theorem 22 (Almost NFL for $\mathbf{m}$ and M_{ot})**.** *Under $\mathbf{m}$, the expected optimisation time grows linearly with $|X|$, regardless of the optimisation strategy.*

Proof. The dominance of $\mathbf{m}$ over u_{NIAH} is used in (7), between an expansion (6) and a contraction (8) according to the definition (1) of performance measures:

$$M^{\mathbf{m}}_{\mathrm{ot},XY}(a) = \sum_{f \in Y^X} \mathbf{m}_{XY}(f) M_{\mathrm{ot},XY}(T^y(a,f)) \quad (6)$$

$$\geq c_{\mathrm{NIAH}} \sum_{f \in Y^X} u_{\mathrm{NIAH},XY}(f) M_{\mathrm{ot},XY}(T^y(a,f)) \quad (7)$$

$$= c_{\mathrm{NIAH}} \cdot M^{u_{\mathrm{NIAH}}}_{\mathrm{ot},XY}(a) \quad (8)$$

Lemma 20 established (8) to be $c_{\mathrm{NIAH}} \cdot (|X|+1)/2$ for all optimisers a. Thus the expected M_{ot}-performance is always bounded below by $c_{\mathrm{NIAH}} \cdot (|X|+1)/2$, which grows linearly with $|X|$. □

Since optimisation time can never grow faster than linearly with $|X|$, the bound is asymptotically tight. In this sense, Theorem 22 may be viewed as an asymptotic almost-NFL theorem for the universal distribution and M_{ot}. The constant c_{NIAH} in the proof may be very small however, so Theorem 22 does not rule out that expected optimisation time differ substantially between optimisers.

8 Conclusion

In this paper we investigated the No Free Lunch theorems when the performance of an algorithm is measured in expectation with respect to Solomonoff's universal distribution. We showed in Theorem 15 that there is a free lunch with respect to this distribution.

Somewhat surprisingly, despite the bias away from randomness exhibited by the universal distribution, the size of the free lunch turns out to be quite small, at least asymptotically (Theorems 18 and 22). The reason for this is that there are many functions that are both simple and hard to optimise. Most notably the needle-in-a-haystack functions, which have complexity of at most $O(\log |X|)$, but for which a maximum cannot be found without $O(|X|)$ probes.

It should be emphasised that there is little need to be too gloomy about the negative results. The upper bounds on the size of the free lunch given in both negative theorems depend on constants that in practise are likely to be very small. Optimisation is a hard problem, so we should not be too surprised if there are some reasonably frequently occurring functions that cannot be efficiently optimised.

The fact that simplicity is not a sufficient characterisation of the difficulty of optimising a function is unfortunate. This is not true in other domains such as supervised learning and sequence prediction where approaches based on Solomonoff's universal prior are theoretically optimal in a certain sense [Hut05]. One difficulty of optimisation lies in the exploration/exploitation problem, which occurs because at each time-step an optimisation algorithm must make a choice between trying to learn the true function and probing the point that it believes to be the maximum.

Since Kolmogorov complexity is (by itself) insufficient for characterising the difficulty of optimising a function, a new criterion is required. We are currently unsure what this should look like and consider this interesting future research.

References

[AT07] Anne Auger and Olivier Teytaud. Continuous lunches are free! In *GECCO'07*, 2007.

[BP06] Yossi Borenstein and Riccardo Poli. Kolmogorov complexity, optimization and hardness. In *CEC'06*, pages 112–119, 2006.

[Cal02] Cristian Calude. *Information and randomness: an algorithmic perspective.* Springer, 2002.

[CO01] Steffen Christensen and Franz Oppacher. What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In *GECCO'01*, pages 1219–1226, 2001.

[DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. Optimization with randomized search heuristics – the (A)NFL Theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002.

[Fre92] Edward Fredkin. Finite nature. *XXVIIth Rencotre de Moriond*, 1992.

[GO05] Evan J Griffiths and Pekka Orponen. Optimisation, block designs and no free lunch theorems. *Information Processing Letters*, 94(2):55–61, 2005.

[GTW+11] Dov M Gabbay, Paul Thagard, John Woods, Prasanta S Bandyopadhyay, and Malcolm R Forster. *Philosophy of Statistics.* Elsevier, 2011.

[Hut05] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Lecture Notes in Artificial Intelligence (LNAI 2167). Springer, 2005.

[Hut07] Marcus Hutter. On Universal Prediction and Bayesian Confirmation. *Theoretical Computer Science*, 384(1):33–48, 2007.

[Hut12] Marcus Hutter. The subjective computable universe. In Hector Zenil, editor, *A Computable Universe: Understanding and Exploring Nature as Computation*, chapter 21, pages 399–416. World Scientific, 2012.

[IT03] Christian Igel and Marc Toussaint. Neutrality and self-adaptation. *Natural Computing*, 2(2):117–132, 2003.

[IT04] Christian Igel and Marc Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3:312–322, 2004.

[Jan13] Thomas Jansen. *Analyzing Evolutionary Algorithms: The Computer Science Perspective.* Springer Berlin Heidelberg, 2013.

[JC11] Pei Jiang and Ying-ping Chen. Free lunches on the discrete Lipschitz class. *Theoretical Computer Science*, 412(17):1614–1628, April 2011.

[LH11] Tor Lattimore and Marcus Hutter. No free lunch versus Occam's razor in supervised learning. In *Proceedings of the Solomonoff 85th Memorial Conference*, Melbourne, Australia, November 2011. Springer.

[LV08] Ming Li and Paul Vitanyi. *Kolmogorov Complexity and its Applications*. Springer Verlag, third edition, 2008.

[McG06] Simon McGregor. No free lunch and algorithmic randomness. In *GECCO'06*, pages 2–4, 2006.

[RH11] Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

[RVW09] Jonathan E Rowe, Michael D Vose, and Alden H Wright. Reinterpreting no free lunch. *Evolutionary computation*, 17(1):117–129, January 2009.

[Str03] Matthew J Streeter. Two broad classes of functions for which a no free lunch result does not hold. In *GECCO'03*, pages 1418–1430, 2003.

[SVW01] Christopher W Schumacher, Michael D Vose, and L Darrell Whitley. The no free lunch and problem description length. In *GECCO'01*, pages 565–570, 2001.

[WM97] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):270–283, 1997.

[Wol02] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[WR06] L Darrell Whitley and Jonathan E Rowe. Subthreshold-seeking local search. *Theoretical Computer Science*, 361(1):2–17, 2006.

Appendix

We here include a proof of Theorem 16. The proof builds on the following definitions and lemmas.

Definition 23. A point $x \in X$ is *incompressible* with respect to the context X, Y if $K(x|X, Y) \geq \log(|X|)$.

At least half of the points of any search space will be incompressible. Functions that only have incompressible maxima (except, possibly, for a maximum at x_1) will play an important role since they are guaranteed to have high complexity. The reason for excluding x_1 will be apparent in the proof of Theorem 16.

Lemma 24. *Let X, Y be a problem context, and let $D \subseteq X$ be a non-empty set of incompressible points. Let $g\colon X \to Y$ have at least one maximum in D, and no maximum outside $D \cup \{x_1\}$. Then $K(g\,|\,X, Y) \geq \log_2(|X|) - c$, where c depends only on the reference machine and not on g, X or Y.*

Proof. Let g be as in the Lemma statement, and let $x_m \in X - \{x_1\}$ be the first maximum of g not at x_1. Then x_m can be coded by means of g with constant length procedure $\textsc{FirstMax}(g)$ that computes the first maximum not at x_1 for a given function g. Hence $K(x_m\,|\,X, Y) \leq K(g\,|\,X, Y) + \ell(\textsc{FirstMax}) + c$. The constant c depends only on the reference machine, and absorbs the cost of initialising $\textsc{FirstMax}$ with a provided description of g.

By assumption x_m was incompressible, so $K(x_m\,|\,X, Y) \geq \log_2|X|$. Combined and rearranged, this gives $K(g\,|\,X, Y) \geq \log_2|X| - \ell(\textsc{FirstMax}) - c$. The lemma now follows by absorbing $\ell(\textsc{FirstMax})$ into c. □

We are now ready for the proof of Theorem 16 that shows that there is free lunch for M_{ot} on the problem $\mathbf{m}$. The key idea is to show that there is a trace after which two unexplored points have different probability of being the maximum.

Theorem 16. *There is free lunch for the problem $\mathbf{m}$ under the performance measure M_{ot} for all problem contexts with sufficiently large search space (the required size depending on the reference machine only).*

Proof. Let $k \geq 2$ and let X,Y be a problem context with $|X| \geq 2k$. Let $D_k \subseteq X$ be of size k and only include incompressible points. Let $Q = X - D_k - \{x_1\}$. Let $G = \{g \in Y^X\colon\ x \in Q \implies g(x) = 0\}$ contain all functions that are 0 on Q. Let f be 0 everywhere, except at x_1, where f is 1. The complexity of f is upper bounded by a constant c_f independent of X. Since $f \in G$, we get $\mathbf{m}_{XY}(G) \geq \mathbf{m}_{XY}(f) \geq 2^{-c_f}$.

Let $x_m \in D_k$ be an incompressible point, and let $G_m = \{g \in G\colon\ g(x_m) = \max g\}$. As the functions in G are all 0 on Q, the cardinality of G_m is at most $|Y|^{|X-Q|} = |Y|^{k+1}$. Also, the functions in G_m all have complexity above $\log|X| - c$ for some c independent of X, Y, by Lemma 24.

We will now show that $\mathbf{m}_{XY}(G_m|G)$ tends to 0 with growing $|X|$, while $\mathbf{m}_{XY}(f|G)$ remains bounded away from 0. This will establish that provided G, a maximum at x_1 is more likely than a maximum at x_m. Provided G, the probability of a maximum at x_1 is always above 2^{-c_f}, since $\mathbf{m}_{XY}(\text{max at } x_1|G) \geq \mathbf{m}_{XY}(f\,|G) \geq \mathbf{m}_{XY}(f) \geq 2^{-c_f}$. A maximum at x_m, on the other hand, is less likely since only functions in G_m can have a maximum there:

$$\begin{aligned} \mathbf{m}_{XY}(\text{max at } x_m|G) &= \mathbf{m}_{XY}(G_m)/\mathbf{m}_{XY}(G) \\ &\leq \mathbf{m}_{XY}(G_m)/2^{-c_f} \\ &= 2^{c_f} \cdot c_{\mathbf{m}_{XY}} \sum_{g \in G_m} 2^{-K(g\,|X,Y)} \end{aligned} \tag{9}$$

Using the lower bound on the complexity from Lemma 24, (9) is bounded by

$$\begin{aligned} &\leq 2^{c_f} \cdot c_{\mathbf{m}_{XY}} \sum_{g \in G_m} 2^{-\log_2|X|+c} \\ &= 2^{c_f} \cdot c_{\mathbf{m}_{XY}} |G_m| \cdot 2^{-\log_2|X|+c} \end{aligned} \tag{10}$$

and since the cardinality of G_1 is less than $|Y|^{k+1}$, (10) is bounded by

$$\begin{aligned} &\leq 2^{c_f} \cdot c_{\mathbf{m}_{XY}} \cdot |Y|^{k+1} \cdot 2^{-\log_2|X|+c} \\ &= \frac{2^{c_f+c} \cdot c_{\mathbf{m}_{XY}} \cdot |Y|^{k+1}}{|X|} \end{aligned} \tag{11}$$

the last equality by elementary simplification.

As $c_{\mathbf{m}_{XY}}$ is bounded above by a constant $c_{\mathbf{m}}$ for all X, Y, (11) goes to 0 with growing search space (and fixed k and Y). This shows that for large enough search spaces, x_1 is more likely to host a maximum than x_m.

Now all that remains is to use this to create two algorithms that perform differently under M_{ot}. Let a start by searching Q in order. If the perceived function points are consistent with f (i.e., the event G is verified), then a proceeds at x_1 and then at x_m, whereafter a searches the remaining points $X - D_k - \{x_1\}$ in order. If the trace is not consistent with f, then a directly proceeds to search all remaining points in order. Define b the same way, with the only exception that after Q it searches x_m before x_1 in case the trace is consistent with f.

This way, a and b will perform the same except when encountering a function in G, in which case a will have a strictly better chance of finding the maximum at step $|Q| + 1$. If neither a nor b finds a maximum at step $|Q| + 1$, then neither x_1 nor x_m is a maximum, so neither a nor b will find a maximum at step $|Q| + 2$ either. Finally, on step $|Q| + 3$ and onwards their behaviour will again be identical, and therefore also their M_{ot} performance. So a has a strictly better chance at step $|Q| + 1$ and a and b's performance is identical on all other steps and in all other situations. This shows that there is a (possibly small) free lunch for M_{ot} on $\mathbf{m}$ for sufficiently large search spaces. □