No Free Lunch versus Occam's Razor in Supervised Learning

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

[1] Australian National University, Canberra, Australia
[2] ETH Zürich, Switzerland
[3] NICTA
{tor.lattimore,marcus.hutter}@anu.edu.au

Abstract. The No Free Lunch theorems are often used to argue that domain specific knowledge is required to design successful algorithms. We use algorithmic information theory to argue the case for a universal bias allowing an algorithm to succeed in all interesting problem domains. Additionally, we give a new algorithm for off-line classification, inspired by Solomonoff induction, with good performance on all structured (compressible) problems under reasonable assumptions. This includes a proof of the efficacy of the well-known heuristic of randomly selecting training data in the hope of reducing the misclassification rate.

Keywords: Supervised learning, Kolmogorov complexity, no free lunch, Occam's razor.

1 Introduction

The No Free Lunch (NFL) theorems, stated and proven in various settings and domains [16,26,27], show that no algorithm performs better than any other when their performance is averaged uniformly over all possible problems of a particular type.[1] These are often cited to argue that algorithms must be designed for a particular domain or style of problem, and that there is no such thing as a general purpose algorithm.

On the other hand, Solomonoff induction [18,19] and the more general AIXI model [9] appear to universally solve the sequence prediction and reinforcement learning problems respectively. The key to the apparent contradiction is that Solomonoff induction and AIXI do not assume that each problem is equally likely. Instead they apply a bias towards more structured problems. This bias is universal in the sense that no class of structured problems is favored over another. This approach is philosophically well justified by Occam's razor.

The two classic domains for NFL theorems are optimisation and classification. In this paper we will examine classification and only remark that the case for optimisation is more complex. This difference is due to the active nature of optimisation where actions affect future observations.

[1] Such results have been less formally discussed long before by Watanabe in 1969 [25].

D.L. Dowe (Ed.): Solomonoff Festschrift, LNAI 7070, pp. 223–235, 2013.

Previously, some authors have argued that the NFL theorems do not disprove the existence of universal algorithms for two reasons.

1. That taking a uniform average is not philosophically the right thing to do, as argued informally in [7].
2. Carroll and Seppi in [1] note that the NFL theorem measures performance as misclassification rate, where as in practise, the utility of a misclassification in one direction may be more costly than another.

We restrict our consideration to the task of minimising the misclassification rate while arguing more formally for a non-uniform prior inspired by Occam's razor and formalised by Kolmogorov complexity. We also show that there exist algorithms (unfortunately only computable in the limit) with very good properties on all structured classification problems.

The paper is structured as follows. First, the required notation is introduced (Section 2). We then state the original NFL theorem, give a brief introduction to Kolmogorov complexity, and show that if a non-uniform prior inspired by Occam's razor is used, then there exists a free lunch (Section 3). Finally, we give a new algorithm inspired by Solomonoff induction with very attractive properties in the classification problem (Section 4).

2 Preliminaries

Here we introduce the required notation and define the problem setup for the No Free Lunch theorems.

Strings. A finite string x over alphabet X is a finite sequence $x_1x_2x_3\cdots x_{n-1}x_n$ with $x_i \in X$. An infinite string x over alphabet X is an infinite sequence $x_1x_2x_3\cdots$. Alphabets are usually countable or finite, while in this paper they will almost always be binary. For finite strings we have a length function defined by $\ell(x) := n$ for $x = x_1x_2\cdots x_n$. The empty string of length 0 is denoted by ϵ. The set X^n is the set of all strings of length n. The set X^* is the set of all finite strings. The set X^∞ is the set of all infinite strings. Let x be a string (finite or infinite) then substrings are denoted $x_{s:t} := x_sx_{s+1}\cdots x_{t-1}x_t$ where $s \leq t$. A useful shorthand is $x_{<t} := x_{1:t-1}$. Let $x, y \in X^*$ and $z \in X^\infty$ with $\ell(x) = n$ and $\ell(y) = m$ then

$$\begin{aligned} xy &:= x_1x_2,\cdots x_{n-1}x_ny_1y_2\cdots y_{m-1}y_m \\ xz &:= x_1x_2,\cdots x_{n-1}x_nz_1z_2z_3\cdots \end{aligned}$$

As expected, xy is finite and has length $\ell(xy) = n + m$ while xz is infinite. For binary strings, we write $\#1(x)$ and $\#0(x)$ to mean the number of 0's and number of 1's in x respectively.

Classification. Informally, a classification problem is the task of matching features to class labels. For example, recognizing handwriting where the features are images and the class labels are letters. In supervised learning, it is (usually) unreasonable to expect this to be possible without any examples of correct classifications. This can be solved by providing a list of feature/class label pairs representing the true classification of each feature. It is hoped that these examples can be used to generalize and correctly classify other features.

The following definitions formalize classification problems, algorithms capable of solving them, as well as the loss incurred by an algorithm when applied to a problem, or set of problems. The setting is that of transductive learning as in [3].

Definition 1 (Classification Problem). *Let X and Y be finite sets representing the feature space and class labels respectively. A classification problem over X, Y is defined by a function $f : X \to Y$ where $f(x)$ is the true class label of feature x.*

In the handwriting example, X might be the set of all images of a particular size and Y would be the set of letters/numbers as well as a special symbol for images that correspond to no letter/number.

Definition 2 (Classification Algorithm). *Let f be a classification problem and $X_m \subseteq X$ be the training features on which f will be known. We write f_{X_m} to represent the function $f_{X_m} : X_m \to Y$ with $f_{X_m}(x) := f(x)$ for all $x \in X_m$. A classification algorithm is a function, A, where $A(f_{X_m}, x)$ is its guess for the class label of feature $x \in X_u := X - X_m$ when given training data f_{X_m}. Note we implicitly assume that X and Y are known to the algorithm.*

Definition 3 (Loss function). *The loss of algorithm A, when applied to classification problem f, with training data X_m is measured by counting the proportion of misclassifications in the testing data, X_u.*

$$L_A(f, X_m) := \frac{1}{|X_u|} \sum_{x \in X_u} [\![A(f_{X_m}, x) \neq f(x)]\!]$$

where $[\![]\!]$ is the indicator function defined by, $[\![expr]\!] = 1$ if expr is true and 0 otherwise.

We are interested in the expected loss of an algorithm on the set of all problems where expectation is taken with respect to some distribution P.

Definition 4 (Expected loss). *Let $\mathcal{M}$ be the set of all functions from X to Y and P be a probability distribution on $\mathcal{M}$. If X_m is the training data then the expected loss of algorithm A is*

$$L_A(P, X_m) := \sum_{f \in \mathcal{M}} P(f) L_A(f, X_m)$$

3 No Free Lunch Theorem

We now use the above notation to give a version of the No Free Lunch Theorem of which Wolpert's is a generalization.

Theorem 1 (No Free Lunch). *Let P be the uniform distribution on $\mathcal{M}$. Then the following holds for any algorithm A and training data $X_m \subseteq X$.*

$$L_A(P, X_m) = |Y - 1|/|Y| \tag{1}$$

The key to the proof is the following observation. Let $x \in X_u$, then for all $y \in Y$, $P(f(x) = y|f|_{X_m}) = P(f(x) = y) = 1/|Y|$. This means no information can be inferred from the training data, which suggests no algorithm can be better than random.

Occam's razor/Kolmogorov Complexity. The theorem above is often used to argue that no general purpose algorithm exists and that focus should be placed on learning in specific domains.

The problem with the result is the underlying assumption that P is uniform, which implies that training data provides no evidence about the true class labels of the test data. For example, if we have classified the sky as blue for the last 1,000 years then a uniform assumption on the possible sky colours over time would indicate that it is just as likely to be green tomorrow as blue, a result that goes against all our intuition.

How then, do we choose a more reasonable prior? Fortunately, this question has already been answered heuristically by experimental scientists who must endlessly choose between one of a number of competing hypotheses. Given any experiment, it is easy to construct a hypothesis that fits the data by using a lookup table. However such hypotheses tend to have poor predictive power compared to a simple alternative that also matches the data. This is known as the principle of parsimony, or Occam's razor, and suggests that simple hypotheses should be given a greater weight than more complex ones.

Until recently, Occam's razor was only an informal heuristic. This changed when Solomonoff, Kolmogorov and Chaitin independently developed the field of algorithmic information theory that allows for a formal definition of Occam's razor. We give a brief overview here, while a more detailed introduction can be found in [13]. An in depth study of the philosophy behind Occam's razor and its formalisation by Kolmogorov complexity can be found in [12,15]. While we believe Kolmogorov complexity is the most foundational formalisation of Occam's razor, there have been other approaches such as MML [23] and MDL [8]. These other techniques have the advantage of being computable (given a computable prior) and so lend themselves to good practical applications.

The idea of Kolmogorov complexity is to assign to each binary string an integer valued *complexity* that represents the length of its shortest description. Those strings with short descriptions are considered simple, while strings with long descriptions are complex. For example, the string consisting of 1,000,000 1's can easily be described as "one million ones". On the other hand, to describe a string generated by tossing a coin 1,000,000 times would likely require a description about 1,000,000 bits long. The key to formalising this intuition is to choose a universal Turing machine as the language of descriptions.

Definition 5 (Kolmogorov Complexity). *Let U be a universal Turing machine and $x \in \mathcal{B}^*$ be a finite binary string. Then define the plain Kolmogorov* *complexity $C(x)$ to be the length of the shortest program (description) p such that $U(p) = x$.*

$$C(x) := \min_{p\in\mathcal{B}^*} \{\ell(p) : U(p) = x\}$$

It is easy to show that C depends on choice of universal Turing machine U only up to a constant independent of x and so it is standard to choose an arbitrary *reference* universal Turing machine.

For technical reasons it is difficult to use C as a prior, so Solomonoff introduced monotone machines to construct the Solomonoff prior, $\mathbf{M}$. A monotone Turing machine has one read-only input tape which may only be read from left to right and one write-only output tape that may only be written to from left to right. It has any number of working tapes. Let T be such a machine and write $T(p) = x$ to mean that after reading p, x is on the output tape. The machines are called monotone because if p is a prefix of q then $T(p)$ is a prefix of $T(q)$. It is possible to show there exists a universal monotone Turing machine U and this is used to define monotone complexity Km and Solomonoff's prior, $\mathbf{M}$.

Definition 6 (Monotone Complexity). *Let U be the reference universal monotone Turing machine then define Km, $\mathbf{M}$ and KM as follows,*

$$\begin{aligned} Km(x) &:= \min\{\ell(p) : U(p) = x*\} \\ \mathbf{M}(x) &:= \sum_{U(p)=x*} 2^{-\ell(p)} \\ KM(x) &:= -\log \mathbf{M}(x) \end{aligned}$$

where $U(p) = x*$ means that when given input p, U outputs x possibly followed by more bits.

Some facts/notes follow.

1. For any n, $\sum_{x\in\mathcal{B}^n} \mathbf{M}(x) \leq 1$.
2. Km, $\mathbf{M}$ and KM are incomputable.
3. $0 < KM(x) \approx Km(x) \approx C(x) < \ell(x) + O(1)$[2]

To illustrate why $\mathbf{M}$ gives greater weight to simple x, suppose x is simple then there exists a relatively short monotone Turing machine p, computing it. Therefore $Km(x)$ is small and so $2^{-Km(x)} \approx \mathbf{M}(x)$ is relatively large.

Since $\mathbf{M}$ is a semi-measure rather than a proper measure, it is not appropriate to use it in place of P when computing expected loss. However it can be normalized to a proper measure, $\mathbf{M}_{norm}$ defined inductively by

$$\mathbf{M}_{norm}(\epsilon) := 1 \qquad \mathbf{M}_{norm}(xb) := \mathbf{M}_{norm}(x)\frac{\mathbf{M}(xb)}{\mathbf{M}(x0) + \mathbf{M}(x1)}$$

[2] The approximation $C(x) \approx Km(x)$ is only accurate to $\log \ell(x)$, while $KM \approx Km$ is almost always very close [5,6]. This is a little surprising since the sum in the definition of $\mathbf{M}$ contains 2^{-Km}. It shows that there are only comparitively few short programs for any x.

Note that this normalisation is not unique, but is philosophically and technically the most attractive and was used and defended by Solomonoff. For a discussion of normalisation, see [13, p.303]. The normalised version satisfies $\sum_{x \in \mathcal{B}^n} \mathbf{M}_{norm}(x) = 1$.

We will also need to define $\mathbf{M}/KM$ with side information, $\mathbf{M}(y; x) := \mathbf{M}(y)$ where $x*$ is provided on a spare tape of the universal Turing machine. Now define $KM(y; x) := -\log \mathbf{M}(y; x)$. This allows us to define the complexity of a function in terms of its output relative to its input.

Definition 7 (Complexity of a function). *Let $X = \{x_1, \cdots, x_n\} \subseteq \mathcal{B}^k$ and $f : X \to \mathcal{B}$ then define the complexity of f, $KM(f; X)$ by*

$$KM(f; X) := KM(f(x_1)f(x_2) \cdots f(x_n); x_1, x_2, \cdots, x_n)$$

An example is useful to illustrate why this is a good measure of the complexity of f.

Example 1. Let $X \subseteq \mathcal{B}^n$ for some n, and $Y = \mathcal{B}$ and $f : X \to Y$ be defined by $f(x) = [\![x_n = 1]\!]$. Now for a complex X, the string $f(x_1)f(x_2) \cdots$ might be difficult to describe, but there is a very short program that can output $f(x_1)f(x_2) \cdots$ when given $x_1 x_2 \cdots$ as input. This gives the expected result that $KM(f; X)$ is very small.

Free Lunch Using Solomonoff Prior. We are now ready to use $\mathbf{M}_{norm}$ as a prior on a problem family. The following proposition shows that when problems are chosen according to the Solomonoff prior that there is a (possibly small) free lunch.

Before the proposition, we remark on problems with maximal complexity, $KM(f; X) = O(|X|)$. In this case f exhibits no structure allowing it to be compressed, which turns out to be equivalent to being random in every intuitive sense [14]. We do not believe such problems are any more interesting than trying to predict random coin flips. Further, the NFL theorems can be used to show that no algorithm can learn the class of random problems by noting that almost all problems are random. Thus a bias towards random problems is not much of a bias (from uniform) at all, and so at most leads to a decreasingly small free lunch as the number of problems increases.

Proposition 1 (Free lunch under Solomonoff prior). *Let $Y = \mathcal{B}$ and fix a $k \in \mathbb{N}$. Now let $X = \mathcal{B}^n$ and $X_m \subset X$ such that $|X_m| = 2^n - k$. For sufficiently large n there exists an algorithm A such that*

$$L_A(\mathbf{M}_{norm}, X_m) < 1/2$$

The proof is omitted due to space limitations, but the idea is very simple. Consider the algorithm such that $A(f|_{X_m}, x) = 1$ if $f(x) = 1$ for all $x \in X_m$ and $A(f|_{X_m}, x)$ is random otherwise. Then show that if the amount of training data is extremely large relative to the testing data then the Solomonoff prior assigns greater weight to the function $f_1(x) := 1$ for all x than the set of functions satisfying $f(x) = 1$ for all $x \in X_m$ but $f(x) \neq 1$ for some $x \in X_u$.

The proposition is unfortunately extremely weak. It is more interesting to know exactly what conditions are required to do much better than random. In the next section we present an algorithm with good performance on all well structured problems when given "good" training data. Without good training data, even assuming a Solomonoff prior, we believe it is unlikely that the best algorithm will perform well.

Note that while it appears intuitively likely that any non-uniform distribution such as $\mathbf{M}_{norm}$ might offer a free lunch, this is in fact not true. It is shown in [17] that there exist non-uniform distributions where the loss over a problem family is independent of algorithm. These distributions satisfy certain symmetry conditions not satisfied by $\mathbf{M}_{norm}$, which allows Proposition 1 to hold.

4 Complexity-Based Classification

Solomonoff induction is well known to solve the online prediction problem where the true value of each classification is known after each guess. In our setup, the true classification is only known for the training data, after which the algorithm no longer receives feedback. While Solomonoff induction can be used to bound the number of total errors while predicting deterministic sequences, it gives no indication of when these errors may occur. For this reason we present a complexity-inspired algorithm with better properties for the offline classification problem.

Before the algorithm we present a little more notation. As usual, let $X = \{x_1, x_2, \cdots, x_n\} \subseteq \mathcal{B}^k$, $Y = \mathcal{B}$ and let $X_m \subseteq X$ be the training data. Now define an indicator function χ by $\chi_i := [\![x_i \in X_m]\!]$.

Definition 8. *Let $f \in Y^X$ be a classification problem. The algorithm A^* is defined in two steps.*

$$\tilde{f} := \operatorname*{arg\,min}_{\tilde{f} \in Y^X} \left\{ KM(\tilde{f}; X) : \chi_i = 1 \implies \tilde{f}(x_i) = f(x_i) \right\}$$

$$A^*(f_{X_m}, x_i) := \tilde{f}(x_i)$$

Essentially A^* chooses for its model the simplest $\tilde{f}$ consistent with the training data and uses this for classifying unseen data. Note that the definition above only uses the value $y_i = f(x_i)$ where $\chi_i = 1$, and so it does not depend on unseen labels.

If $KM(f; X)$ is "small" then the function we wish to learn is simple so we should expect to be able to perform good classification, even given a relatively small amount of training data. This turns out to be true, but only with a good choice of training data. It is well known that training data should be "broad enough", and this is backed up by the example below and by Theorem 2, which give an excellent justification for random training data based on good theoretical

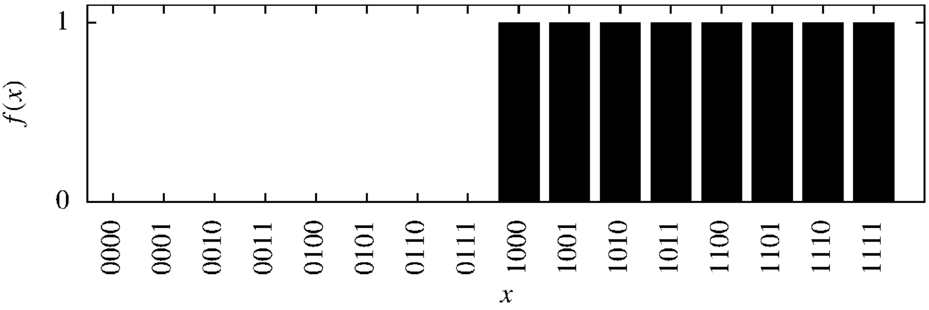

Fig. 1. A simple problem

(Theorem 2) and philosophical (AIT) underpinnings. The following example demonstrates the effect of bad training data on the performance of A^*.

Example 2. Let $X = \{0000, 0001, 0010, 0011, \cdots, 1101, 1110, 1111\}$ and $f(x)$ be defined to be the first bit of x as in Figure 1. Now suppose $\chi = 1^8 0^8$ (So the algorithm is only allowed to see the true class labels of x_1 through x_8). In this case, the simplest $\tilde{f}$ consistent with the first 16 data points, all of which are zeros, is likely to be $\tilde{f}(x) = 0$ for all $x \in X$ and so A^* will fail on every piece of testing data!

On the other hand, if $\chi = 0010100111011 01$, which was generated by tossing a coin 16 times, then $\tilde{f}$ will very likely be equal to f and so A^* will make no errors. Even if χ is zero about the critical point in the middle ($\chi_8 = \chi_9 = 0$) then $\tilde{f}$ should still match f mostly around the left and right and will only be unsure near the middle.

Note, the above is not precisely true since for small strings the dependence of *KM* on the universal monotone Turing machine can be fairly large. However if we increase the size of the example so that $|X| > 1000$ then these quirks disappear for natural reference universal Turing machines.

Definition 9 (Entropy). *Let* $\theta \in [0, 1]$

$$H(\theta) := \begin{cases} -[\theta \log \theta + (1-\theta)\log(1-\theta)] & \textit{if } \theta \neq 0 \textit{ and } \theta \neq 1 \\ 0 & \textit{otherwise} \end{cases}$$

Theorem 2. *Let* $\theta \in (0, 1)$ *be the proportion of data to be given for training then:*

1. *There exists a* $\chi \in \mathcal{B}^\infty$ *(training set) such that for all* $n \in \mathbb{N}$, $\theta n - c_1 < \#1(\chi_{1:n}) < \theta n + c_1$ *and* $nH(\theta) - c_2 < KM(\chi_{1:n})$ *for some* $c_1, c_2 \in \mathbb{R}^+$.
2. *For* $n = |X|$, *the loss of algorithm* A^* *when using training data determined by* χ *is bounded by*

$$L_{A^*}(f, X_m) < \frac{2KM(f; X) + KM(X) + c_2 + c_3}{n(1 - \theta - c_1/n)\log(1 - \theta + c_1/n)^{-1}}$$

where c_3 *is some constant independent of all inputs.*

This theorem shows that A^* will do well on all problems satisfying $KM(f;X) = o(n)$ when given good (but not necessarily a lot) of training data. Before the proof, some remarks.

1. The bound is a little messy, but for small θ, large n and simple X we get $L_{A^*}(f, X_m) \overset{\approx}{<} 2KM(f;X)/(n\theta)$.
2. The loss bound is extremely bad for large θ. We consider this unimportant since we only really care if θ is small. Also, note that if θ is large then the number of points we have to classify is small and so we still make only a few mistakes.
3. The constants c_1, c_2 and c_3 are relatively small (around 100-500). They represent the length of the shortest programs computing simple transformations or encodings. This *is* dependent on the universal Turing machine used to define the Solomonoff distribution, but for a *natural* universal Turing machine we expect it to be fairly small [9, sec.2.2.2].
4. The "special" χ is not actually that special at all. In fact, it can be generated easily with probability 1 by tossing a coin with bias θ infinitely often. More formally, it is a μ Martin-Löf random string where $\mu(1|x) = \theta$ for all x. Such strings form a μ-measure 1 set in $\mathcal{B}^\infty$.

Proof (Theorem 2). The first is a basic result in algorithmic information theory [13, p.318]. Essentially choosing χ to be Martin-Löf random with respect to a Bernoulli process parameterized by θ. From now on, let $\bar{\theta} = \#1(\chi)/n$. For simplicity we write $x := x_1x_2\cdots x_n$, $y := f(x_1)f(x_2)\cdots f(x_n)$, and $\tilde{y} := \tilde{f}(x_1)\tilde{f}(x_2)\cdots\tilde{f}(x_n)$. Define indicator ψ by $\psi_i := [\![\chi_i = 0 \wedge y_i = \tilde{y}_i]\!]$. Now note that there exists $c_3 \in \mathbb{R}$ such that

$$KM(\chi_{1:n}) < KM(\psi_{1:n}; y, \tilde{y}) + KM(y;x) + KM(\tilde{y};x) + KM(x) + c_3 \tag{2}$$

This follows since we can easily use y, $\tilde{y}$ and $\psi_{1:n}$ to recover $\chi_{1:n}$ by $\chi_i = 1$ if and only if $y_i = \tilde{y}_i$ and $\psi_i \neq 1$. The constant c_3 is the length of the reconstruction program. Now $KM(\tilde{y};x) \leq KM(y;x)$ follows directly from the definition of $\tilde{f}$. We now compute an upper bound on $KM(\psi)$. Let $\alpha := L_{A^*}(f, X_m)$ be the proportion of the testing data on which A^* makes an error. The following is easy to verify:

1. $\#1(\psi) = (1-\alpha)(1-\bar{\theta})n$
2. $\#0(\psi) = (1-(1-\alpha)(1-\bar{\theta}))n$
3. $y_i \neq \tilde{y}_i \implies \psi_i = 0$
4. $\#1(y \oplus \tilde{y}) = \alpha(1-\bar{\theta})n$ where $\oplus$ is the exclusive or function.

We can use point 3 above to trivially encode ψ_i when $\tilde{y}_i \neq y_i$. Aside from these, there are exactly $\bar{\theta}n$ 0's and $(1-\alpha)(1-\bar{\theta})n$ 1's. Coding this subsequence using frequency estimation gives a code for $\psi_{1:n}$ given y and $\tilde{y}$, which we substitute into (2).

$$\begin{aligned} nH(\bar{\theta}) - c_2 \leq KM(\chi_{1:n}) &\leq KM(\psi_{1:n}; y, \tilde{y}) + KM(y;x) + KM(\tilde{y};x) \\ &\quad + KM(x) + c_3 \\ &\leq 2KM(y;x) + KM(x) + nJ(\bar{\theta}, \alpha) + c_3 \end{aligned} \tag{3}$$

where $J(\bar{\theta}, \alpha) := \left[\bar{\theta} + (1-\bar{\theta})(1-\alpha)\right] H\left(\bar{\theta} / \left[\bar{\theta} + (1-\bar{\theta})(1-\alpha)\right]\right)$. An easy technical result (Lemma 1 in the appendix) shows that for $\bar{\theta} \in (0,1)$

$$0 \leq \alpha(1-\bar{\theta}) \log \frac{1}{1-\bar{\theta}} \leq H(\bar{\theta}) - J(\bar{\theta}, \alpha)$$

Therefore $n\alpha(1-\bar{\theta}) \log \frac{1}{1-\bar{\theta}} \leq 2KM(y;x) + KM(x) + c_2 + c_3$. The result follows by rearranging and using part 1 of the theorem. □

Since the features are known, it is unexpected for the bound to depend on their complexity, $KM(X)$. Therefore it is not surprising that this dependence can be removed at a small cost, and with a little extra effort.

Theorem 3. *Under the same conditions as Theorem 2, the loss of A^* is bounded by*

$$L_{A^*}(f, X_m) < \frac{2KM(f;X) + 2\left[\log|X| + \log\log|X|\right] + c}{n(1-\theta-c_1/n)\log(1-\theta+c_1/n)^{-1}}$$

where c is some constant independent of inputs.

This version will be preferred to Theorem 2 in cases where $KM(X) > 2$ $[\log|X| + \log\log|X|]$. The proof of Theorem 3 is almost identical to that of Theorem 2.
Proof sketch: The idea is to replace equation (2) by

$$KM(\chi_{1:n}, x) < KM(\psi_{1:n}; y, \tilde{y}) + KM(y;x) + KM(\tilde{y};x) + KM(x) + c_3 \qquad (4)$$

Then use the following identities $K(\chi_{1:n}; x, K(x)) + K(x) < K(\chi_{1:n}, x) – K(\ell(x)) < KM(\chi_{1:n}, x)$ where the inequalities are true up to constants independent of x and χ. Next a counting argument in combination with Stirling's approximation can be used to show that for most χ satisfying the conditions in Theorem 2 have $KM(\chi_{1:n}) < K(\chi_{1:n}) < K(\chi_{1:n}; x, K(x)) + \log \ell(x) + r$ for some constant $r > 0$ independent of x and χ. Finally use $KM(x) < K(x)$ for all x and $K(\ell(x)) < \log \ell(x) + 2\log\log \ell(x) + r$ for some constant $r > 0$ independent of x to rearrange (4) into

$$\begin{aligned} KM(\chi_{1:n}) < KM(\psi_{1:n}; y, \tilde{y}) + KM(y;x) + KM(\tilde{y};x) + 2\log \ell(x) \\ + 2\log\log \ell(x) + c \end{aligned}$$

for some constant $c > 0$ independent of χ, ψ, x and y. Finally use the techniques in the proof of Theorem 2 to complete the proof. □

5 Discussion

Summary. Proposition 1 shows that if problems are distributed according to their complexity, as Occam's razor suggests they should, then a (possibly small) free lunch exists. While the assumption of simplicity still represents a bias towards certain problems, it is a universal one in the sense that no style of structured problem is more favoured than another.

In Section 4 we gave a complexity-based classification algorithm and proved the following properties:

1. It performs well on problems that exhibit some compressible structure, $KM(f;X) = o(n)$.
2. Increasing the amount of training data decreases the error.
3. It performs better when given a good (broad/randomized) selection of training data.

Theorem 2 is reminiscent of the transductive learning bounds of Vapnik and others [3,20,21], but holds for *all* Martin-Löf random training data, rather than with high probability. This is different to the predictive result in Solomonoff induction where results hold with probability 1 rather than for all Martin-Löf random sequences [11]. If we assume the training set is sampled randomly, then our bounds are comparable to those in [3].

Unfortunately, the algorithm of Section 4 is incomputable. However Kolmogorov complexity can be approximated via standard compression algorithms, which may allow for a computable approximation of the classifier of Section 4. Such approximations have had some success in other areas of AI, including general reinforcement learning [22] and unsupervised clustering [2].

Occam's razor is often thought of as the principle of choosing the simplest hypothesis matching your data. Our definition of simplest is the hypothesis that minimises $KM(f;X)$ (maximises $M(f;X)$). This is perhaps not entirely natural from the informal statement of Occam's razor, since $M(x)$ contains contributions from all programs computing x, not just the shortest. We justify this by combining Occam's razor with Epicurus principle of multiple explanations that argues for all consistent hypotheses to be considered. In some ways this is the most natural interpretation as no scientist would entirely rule out a hypothesis just because it is slightly more complex than the simplest. A more general discussion of this issue can be found in [4, sec.4]. Additionally, we can argue mathematically that since $KM \approx Km$, the simplest hypothesis is very close to the mixture.[3] Therefore the debate is more philosophical than practical in this setting.

An alternative approach to formalising Occam's razor has been considered in MML [23]. However, in the deterministic setting the probability of the data given the hypothesis satisfies $P(D|H) = 1$. This means the two part code reduces to the code-length of the prior, $\log(1/P(H))$. This means the hypothesis with minimum message length depends only on the choice of prior, not the complexity of coding the data. The question then is how to choose the prior, on which MML gives no general guidance. Some discussion of Occam's razor from a Kolmogorov complexity viewpoint can be found in [10,12,15], while the relation between MML and Kolmogorov complexity is explored in [24].

Assumptions. We assumed finite X, Y, and deterministic f, which is the standard transductive learning setting. Generalisations to countable spaces may still be possible using complexity approaches, but non-computable real numbers prove more difficult. One can either argue by the strong Church-Turing thesis

[3] The bounds of Section 4 would depend on the choice of complexity at most logarithmically in $|X|$ with KM providing the uniformly better bound.

that non-computable reals do not exist, or approximate them arbitrarily well. Stochastic f are interesting and we believe a complexity-based approach will still be effective, although the theorems and proofs may turn out to be somewhat different.

Acknowledgements. We thank Wen Shao and reviewers for valuable feedback on earlier drafts and the Australian Research Council for support under grant DP0988049.

References

1. Carroll, J., Seppi, K.: No-free-lunch and Bayesian optimality. In: IJCNN Workshop on Meta-Learning (2007)
2. Cilibrasi, R., Vitanyi, P.: Clustering by compression. IEEE Transactions on Information Theory 51(4), 1523–1545 (2005)
3. Derbeko, P., El-yaniv, R., Meir, R.: Error bounds for transductive learning via compression and clustering. In: NIPS, vol. 16 (2004)
4. Dowe, D.: MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness. In: Handbook of Philosophy of Statistics, vol. 7, pp. 901–982. Elsevier (2011)
5. Gács, P.: On the relation between descriptional complexity and algorithmic probability. Theoretical Computer Science 22(1-2), 71–93 (1983)
6. Gács, P.: Expanded and improved proof of the relation between description complexity and algorithmic probability (2008) (unpublished)
7. Giraud-Carrier, C., Provost, F.: Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In: ICML Workshop on Meta-Learning, pp. 9–16 (2005)
8. Grünwald, P.: The Minimum Description Length Principle. MIT Press Books, vol. 1. The MIT Press (2007)
9. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004)
10. Hutter, M.: A complete theory of everything (will be subjective). Algorithms 3(4), 329–350 (2010)
11. Hutter, M., Muchnik, A.: On semimeasures predicting Martin-Löf random sequences. Theoretical Computer Science 382(3), 247–261 (2007)
12. Kirchherr, W., Li, M., Vitanyi, P.: The miraculous universal distribution. The Mathematical Intelligencer 19(4), 7–15 (1997)
13. Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer (2008)
14. Martin-Löf, P.: The definition of random sequences. Information and Control 9(6), 602–619 (1966)
15. Rathmanner, S., Hutter, M.: A philosophical treatise of universal induction. Entropy 13(6), 1076–1136 (2011)
16. Schaffer, C.: A conservation law for generalization performance. In: Proceedings of the Eleventh International Conference on Machine Learning, pp. 259–265. Morgan Kaufmann (1994)
17. Schumacher, C., Vose, M., Whitley, L.: The no free lunch and problem description length. In: Spector, L., Goodman, E.D. (eds.) GECCO 2001: Proc. of the Genetic and Evolutionary Computation Conf., pp. 565–570. Morgan Kaufmann, San Francisco (2001)

18. Solomonoff, R.: A formal theory of inductive inference, Part I. Information and Control 7(1), 1–22 (1964)
19. Solomonoff, R.: A formal theory of inductive inference, Part II. Information and Control 7(2), 224–254 (1964)
20. Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer, New York (1982)
21. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd edn. Springer, Berlin (2000)
22. Veness, J., Ng, K.S., Hutter, M., Uther, W., Silver, D.: A Monte Carlo AIXI approximation. Journal of Artificial Intelligence Research 40, 95–142 (2011)
23. Wallace, C., Boulton, D.: An information measure for classification. The Computer Journal 11(2), 185–194 (1968)
24. Wallace, C., Dowe, D.: Minimum message length and Kolmogorov complexity. The Computer Journal 42(4), 270–283 (1999)
25. Watanabe, S., Donovan, S.: Knowing and guessing; a quantitative study of inference and information. Wiley, New York (1969)
26. Wolpert, D.: The supervised learning no-free-lunch theorems. In: Proc. 6th Online World Conference on Soft Computing in Industrial Applications, pp. 25–42 (2001)
27. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

A Technical Proofs

Lemma 1 (Entropy inequality).

$$\begin{aligned} 0 &\leq \alpha(1-\theta)\log\frac{1}{1-\theta} && (5)\\ &\leq H(\theta) - [\theta + (1-\theta)(1-\alpha)]\, H\left(\frac{\theta}{\theta+(1-\theta)(1-\alpha)}\right) && (6)\end{aligned}$$

With equality only if $\theta \in \{0,1\}$ or $\alpha = 0$

Proof. First, (5) is trivial. To prove (6), note that for $\alpha = 0$ or $\theta \in \{0,1\}$, equality is obvious. Now, fixing $\theta \in (0,1)$ and computing.

$$\begin{aligned} &\frac{\partial}{\partial\alpha}\left[H(\theta) - [\theta + (1-\theta)(1-\alpha)]\, H\left(\frac{\theta}{\theta+(1-\theta)(1-\alpha)}\right)\right]\\ &= (1-\theta)\log\frac{1-\alpha(1-\theta)}{(1-\alpha)(1-\theta)}\\ &\geq (1-\theta)\log(1-\theta)^{-1}\end{aligned}$$

Therefore integrating both sides over α gives,

$$\alpha(1-\theta)\log(1-\theta)^{-1} \leq H(\theta) - [\theta + (1-\theta)(1-\alpha)]\, H\left(\frac{\theta}{\theta+(1-\theta)(1-\alpha)}\right)$$

as required. □