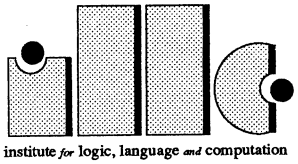

MING LEE, JOHN TROMP,
PAUL VITÁNYI

Sharpening Occam's Razor
(extended abstract)

CT-94-03, *received*: April 1994

ILLC Research Report and Technical Notes Series
Series editor: Dick de Jongh

Computation and Complexity Theory (CT) Series, ISSN: 0928-3323

Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Plantage Muidergracht 24
NL-1018 TV Amsterdam
The Netherlands
e-mail: illc@fwi.uva.nl

Sharpening Occam's Razor

(Extended Abstract)

Ming Li* Waterloo

John Tromp† Waterloo

Paul Vitányi‡ CWI/UvA

Abstract

We provide a new representation-independent formulation of Occam's razor theorem, based on Kolmogorov complexity. This new formulation allows us to:

- Obtain better sample complexity than both length-based [4] and VC-based [3] versions of Occam's razor theorem, in many applications.
- Achieve a sharper reverse of Occam's razor theorem than that of [5]. Specifically, we weaken the assumptions made in [5] and extend the reverse to superpolynomial running times.

1 Introduction

Occam's razor theorem as formulated by [3, 4] is arguably the substance of efficient pac learning. Roughly speaking, it says that in order to learn, it suffices to compress. A partial reverse, showing the necessity of compression, has been proved by Board and Pitt [5]. Since the theorem is about the relation between effective compression and pac learning, it is natural to assume that a sharper version ensues by couching it in terms of the *ultimate* limit to effective compression which is the Kolmogorov complexity. We present results in that direction.

Despite abundant research generated by its importance, several aspects of Occam's razor theorem remain unclear. There are basically two versions. The VC dimension-based version (Theorem

*Supported in part by the NSERC Operating Grant OGP0046506 and ITRC. Address: Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. E-mail: mli@math.uwaterloo.ca

†This work was supported in part by the NSERC International Fellowship and ITRC. Address: Department of Computer Science, University of Waterloo, Ont. N2L 3G1, Canada. E-mail: tromp@math.uwaterloo.ca

‡Partially supported by the European Community through NeuroCOLT ESPRIT Working Group Nr. 8556, and by NWO through NFI Project ALADDIN under Contract number NF 62-376. Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. E-mail: paulv@cwi.nl

3.1.1 of [3]) gives the following upper bound on sample complexity: For a hypothesis space H with $VCdim(H) = d$, $1 \leq d < \infty$,

$$m(H, \delta, \epsilon) \leq \frac{4}{\epsilon}(d \log \frac{12}{\epsilon} + \log \frac{2}{\delta}). \tag{1}$$

The following lower bound was proved by Ehrenfeucht *et al* [6].

$$m(H, \delta, \epsilon) > \max(\frac{d-1}{32\epsilon}, \frac{1}{\epsilon} \ln \frac{1}{\delta}). \tag{2}$$

The upper bound in Equation 1 and the lower bound in Equation 2 differ by a factor $\Theta(\log \frac{1}{\epsilon})$. It was show in [8] that this factor is, in a sense, unavoidable.

When H is finite, one can directly obtain the following bound on sample complexity for any consistent algorithm:

$$m(H, \delta, \epsilon) \leq \frac{1}{\epsilon} \ln \frac{|H|}{\delta}. \tag{3}$$

For a graded boolean space H_n, we have the following relationship between the VC dimension d of H_n and the cardinality of H_n,

$$d \leq \log |H_n| \leq nd. \tag{4}$$

When $\log |H_n| = O(d)$ holds, then the sample complexity upper bound given by Equation 3 can be seen to match the lower bound of Equation 2 up to a constant factor, and thus any consistent algorithm achieves optimal sample complexity for such hypothesis spaces.

The length based version of Occam's razor gives the sample complexity, for given ϵ and δ:

$$m = \max(\frac{2}{\epsilon} \ln \frac{1}{\delta}, (\frac{(2 \ln 2) s^{\beta}}{\epsilon})^{1/(1-\alpha)})), \tag{5}$$

when the *deterministic* occam algorithm returns a consistent hypothesis of length at most $m^{\alpha} s^{\beta}$ with $\alpha < 1$ and s is the length of the target concept.

In summary, the VC dimension based occam's razor theorem may be hard to use and it sometimes does not give the best sample complexity. The length-based Occam's razor is more convenient to use and often gives better sample complexity in the discrete case.

However, as we will demonstrate in this paper, the fact that the length-based Occam's razor theorem sometimes gives inferior sample complexity, can be due to the redundant representation format of the concept.

We believe Occam's razor theorem should be "representation-independent". That is, it should not be dependent on accidents of "representation format". (See [13] for other representation-independence issues.) In fact, the sample complexities given in Equations 1 and 3 are indeed representation-independent. However they are not easy to use and do not give optimal sample complexity.

In this paper, we give a Kolmogorov complexity-based Occam's razor theorem. We will demonstrate that our KC-based Occam's razor theorem is convenient to use (as convenient as the length based version), gives a better sample complexity than the length based version, and is representation-independent. In fact, the length based version can be considered as a specific computable approximation to the KC-based Occam's razor.

As one of the examples, we will demonstrate that the standard trivial learning algorithm for monomials actually often has a *better sample complexity* than the more sophisticated Haussler's greedy algorithm [7], using our KC-based Occam's razor theorem. This is contrary to the common belief that Haussler's algorithm is better.

Another issue related to Occam's razor theorem is the status of the reverse assertion. Although a partial reverse of Occam's razor theorem has been proved by [5], it applied only to the case of polynomial running time and sample complexity. They also required a property of closure under exception list. This latter requirement, although quite general, excludes some reasonable concept classes. Our new formulation of Occam's razor theorem allows us to prove a more general reverse of Occam's razor theorem, allowing the arbitrary running time and weakening the requirement of exception list of [5].

2 Occam's Razor

Let us assume the usual definitions, say Anthony and Biggs [1]. Also assume the notation of Board and Pitt [5]. For Kolmogorov complexity we assume the basics of [11].

In the following Σ is a finite alphabet. i.e. we consider only discrete learning problems in this paper.

First we define a pac-algorithm and a generalized notion of Occam-algorithm.

Definition 1 *A* pac-algorithm *for a class of representations* $\mathbf{R} = (R, \Gamma, c, \Sigma)$ *is a randomized algorithm* L *such that, for any* $s, n \geq 1, 0 < \epsilon, \delta < 1, r \in R^{\leq s}$, *and any probability distribution* D *on* $\Sigma^{\leq n}$, *if* L *is given* s, n, ϵ, δ *as input and has access to an oracle providing examples of* $c(r)$ *according to* D, *then* L, *with probability at least* $1 - \delta$, *outputs a representation* r' *such that* $D(r' \oplus r) \leq \epsilon$. *The running time and sample complexity of the pac-algorithm are expressed as functions* $t(n, s, \epsilon, \delta)$ *and* $m(n, s, \epsilon, \delta)$.

Definition 2 *An* Occam-algorithm *for a class of representations* $\mathbf{R} = (R, \Gamma, c, \Sigma)$ *is a randomized algorithm which on input of a sample of length* m *of* $r \in R$, *and any* $\gamma > 0$, *with probability at least* $1-\gamma$ *outputs a representation* r' *consistent with the sample, such that* $K(r'|r, n, s) = m/f(m, n, s, \gamma)$, *with* $f(m, n, s, \gamma)$, *the compression achieved, being an increasing function of* m. *The running time*

of the Occam-algorithm is expressed as a function $t(m, n, s, \gamma)$, where n is the maximum length of the input examples.

Our first theorem is a Kolmogorov complexity based Occam's Razor. We denote the minimum m such that $f(m, n, s, \gamma) \geq x$ by $f^{-1}(x, n, s, \gamma)$.

Theorem 1 *Suppose we have an Occam-algorithm for $\mathbf{R} = (R, \Gamma, c, \Sigma)$ with compression $f(m, n, s, \gamma)$. Write f as $f(m, \gamma)$ with the other parameters implicit. Then there is a pac-learning algorithm for $\mathbf{R}$ with sample complexity*

$$m(n, s, \epsilon, \delta) = \max(\frac{2}{\epsilon} \ln \frac{2}{\delta}, f^{-1}(\frac{2 \ln 2}{\epsilon}, \delta/2)),$$

and running time $t(n, s, \epsilon, \delta) = t(m(n, s, \epsilon, \delta), n, s, \delta/2)$.

Proof. On input of ϵ, δ, s, n, the learning algorithm will take a sample of length $m = m(n, s, \epsilon, \delta)$ from the oracle, then use the Occam algorithm with $\gamma = \delta/2$ to find a hypothesis (with probability at least $1 - \delta/2$) consistent with the sample and with low Kolmogorov complexity. In the proof we further abbreviate f to $f(m)$ with the other parameters implicit. Learnability follows in the standard manner from bounding (by the remaining $\delta/2$) the probability that all m examples of the target concept fall within the, probability ϵ or greater, symmetric difference with a bad hypothesis. Let $m \geq m(n, s, \epsilon, \delta)$. Then $m \geq f^{-1}(\frac{2 \ln 2}{\epsilon}, \frac{\delta}{2})$ gives

$$\epsilon - \frac{\ln 2}{f(m)} \geq \frac{\epsilon}{2}.$$

Bounding $(1 - \epsilon)^m$ by $e^{-\epsilon m}$ and taking negative logarithms,

$$2^{m/f(m)}(1 - \epsilon)^m \leq \delta/2 \Leftrightarrow$$

$$m(\epsilon - \frac{\ln 2}{f(m)}) \geq \ln \frac{2}{\delta},$$

which follows from the above and the first lower bound on m. □

Corollary 2 *When the compression is of the form*

$$f(m, n, s, \gamma) = \frac{m^{1-\alpha}}{p(n, s, \gamma)},$$

one can achieve a sample complexity of

$$\max(\frac{2}{\epsilon} \ln \frac{2}{\delta}, \left(\frac{(2 \ln 2) p(n, s, \delta/2)}{\epsilon}\right)^{1/(1-\alpha)}).$$

In the special case of total compression, where $\alpha = 0$, this further reduces to

$$\frac{2}{\epsilon}(\max(\ln\frac{2}{\delta}, (\ln 2)p(n,s,\delta/2))). \tag{6}$$

For deterministic Occam-algorithms, we can furthermore replace $2/\delta$ and $\delta/2$ in Theorem 1 by $1/\delta$ and δ respectively.

Remark. Essentially, our new Kolmogorov complexity condition is a computationally universal generalization of the length condition in the original Occam's razor theorem of [4]. Here, in Theorem 1, we consider the shortest description length over all effective representations. This is representation-independent in the very strong sense of being an absolute and objective notion, which is recursively invariant by Church's thesis and the ability of universal machines to simulate each another.

Definition 3 *An* exception handler *for a class of representations $\mathbf{R} = (R, \Gamma, c, \Sigma)$ is an algorithm which on input of a representation $r \in R$ of length s, and an $x \in \Sigma^*$ of length n, outputs a representation r' of the concept $c(r) \oplus \{x\}$, of length at most $e(s,n)$, where e is the exception expansion function. The running time of the exception-handler is expressed as a function $t(n,s)$ of the representation and exception lengths. If $t(n,s)$ is polynomial in n, s and $e(s,n)$ is of the form $s + p(n)$ for some polynomial $p()$ then we say $\mathbf{R}$ is* polynomially closed under exceptions.

Theorem 3 *Let L be a pac-algorithm and E be an exception handler for $\mathbf{R} = (R, \Gamma, c, \Sigma)$. Then there is an Occam algorithm for $\mathbf{R}$ with compression $\frac{1}{2\epsilon n}$, where ϵ, depending on m, n, s, γ, is such that $m(n,s,\epsilon,\gamma) = \epsilon m$ holds.*

Proof. The proof is obtained in a fashion similar to Board and Pitt. Suppose we are given a sample of length m and confidence parameter γ. Assume without loss of generality that the sample contains m different examples. Define a uniform distribution on these examples with $\mu(x) = 1/m$ for each x in the sample. Let ϵ be as described. E.g. when $m(n,s,\epsilon,\gamma) = (\frac{1}{\epsilon})^b$ for some constant b, then $\epsilon = m^{-1/(b+1)}$. Apply L with $\delta = \gamma$ and above ϵ. It produces a concept which is correct with error ϵ, giving up to ϵm exceptions. We can just add these one by one using the exception handler. This will expand the concept size, but not the Kolmogorov complexity. The resulting representation can be described by the examples used plus the exceptions found, each taking n bits. This gives the claimed compression. □

Definition 4 *A* majority-of-3 algorithm *for a class of representations $\mathbf{R} = (R, \Gamma, c, \Sigma)$ is an algorithm which on input of 3 representation $r_1, r_2, r_3 \in R^{\leq s}$, outputs a representation r' of the concept $MAJ(r_1, r_2, r_3)$ of length at most $e(s)$, where e is the majority expansion function. The running time of the algorithm is expressed as a function $t(s)$ of the maximum representation length. If $t(s)$ and $e(s)$ are polynomial in s then we say $\mathbf{R}$ is* polynomially closed under majority-of-3.

Theorem 4 *Let L be a pac-algorithm with sample complexity $m(n, s, \epsilon, \delta)$ subquadratic in $\frac{1}{\epsilon}$, and let M be a majority-of-3 algorithm for $\mathbf{R} = (R, \Gamma, c, \Sigma)$. Then there is an Occam algorithm for $\mathbf{R}$ with compression $m/3nm(n, s, \frac{1}{2\sqrt{m}}, \gamma/3)$.*

Proof. Let us be given a sample of length m. Take $\delta = \gamma/3$ and $\epsilon = 1/\sqrt{m}$.

Stage 1: Define a uniform distribution on the m examples with $\mu(x) = 1/m$ for each x in the sample. Apply the learning algorithm. It produces (with probability at least $1 - \gamma/3$) a hypothesis r_1 which has error less than ϵ, giving up to $\epsilon m = \sqrt{m}$ exceptions. Denote this set of exceptions by E_1.

Stage 2: Define a new distribution on the m examples with $\mu(x) = \epsilon_2 = 1/(2\sqrt{m})$ for each x in E_1, and $\mu(x) = (1 - |E_1|/2\sqrt{m})/(m - |E_1|)$ for each x not in E_1. Apply the learning algorithm with error bound ϵ_2. It produces (with probability at least $1 - \gamma/3$) a hypothesis r_2 which is correct on all of E_1 and with error less than ϵ_2 on the remaining examples. This gives up to $\epsilon_2(m - |E_1|)/(1 - |E_1|/2\sqrt{m}) < \sqrt{m}$ exceptions. Denote this set E_2. We have that E_2 is disjoint from E_1.

Stage 3: Define a new distribution on the m examples with $\mu(x) = 1/|E_1 \cup E_2|$ for each x in $E_1 \cup E_2$, and $\mu(x) = 0$ elsewhere. Apply the learning algorithm with error bound $\epsilon_3 = 1/2\sqrt{m}$. Note that $|E_1| \leq \sqrt{m}$ and $E_2 < \sqrt{m}$ gives that for x in $E_1 \cup E_2$, $\mu(x) > \epsilon_3$. Thus the algorithm produces (with probability at least $1 - \gamma/3$) a hypothesis r_3 which is correct on all of E_1 and E_2 and which might be totally wrong elsewhere (we don't care).

In total the number of examples consumed by the pac-algorithm is at most $3m(n, s, \frac{1}{2\sqrt{m}}, \gamma/3)$ each requiring n bits to describe. The three representations are combined into one representing the majority of the 3 concepts. This is necessarily correct on all of the m examples, since the 3 exception-sets are all disjoint. Furthermore, it can be described in terms of the examples fed to the pac-algorithm and thus achieves compression $f(m, n, s, \gamma) = m/3nm(n, s, \frac{1}{2\sqrt{m}}, \gamma/3)$. This is seen to be an increasing function of m given the assumed subquadratic sample complexity. □

The following corollaries use the fact that if a class is learnable, it must have finite VC-dimension and hence, according to Equation 1, they are learnable with sample complexity subquadratic in $\frac{1}{\epsilon}$.

Corollary 5 *Let a class $\mathbf{R} = (R, \Gamma, c, \Sigma)$ be closed under either exceptions or majority-of-3. Then $\mathbf{R}$ is pac-learnable iff there is an Occam algorithm for $\mathbf{R}$.*

Corollary 6 *Let a class $\mathbf{R} = (R, \Gamma, c, \Sigma)$ be polynomially closed under either exceptions or majority-of-3. Then $\mathbf{R}$ is polynomially pac-learnable iff there is a polynomial time Occam algorithm for $\mathbf{R}$.*

Example. Consider threshold circuits, acyclic circuits whose nodes compute threshold functions of the form $a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq \delta$, $x_i \in \{0, 1\}, a_i, \delta \in N$ (note that no expressive power

is gained by allowing rational weights and threshold). A simple way of representing circuits over the binary alphabet is to number each node and use *prefix-free encodings* of these numbers. For instance, encode i as $1^{|\text{bin}(i)|}0\text{bin}(i)$, the binary representation of i preceded by its length in unary. A complete node encoding then consists of the encoded index, encoded weights, threshold, encoded degree, and encoded indices of the nodes corresponding to its inputs. A complete circuit can be encoded with a node-count followed by a sequence of node-encodings. For this representation, a majority-of-3 algorithm is easily constructed that renumbers two of its three input representations, and combines the three by adding a 3-input node computing the majority function $x_1+x_2+x_3 \geq 2$. It is clear that under this representation, the class of threshold circuits are polynomially closed under majority-of-3. On the other hand they are not closed under exceptions, or under the exception lists of Board and Pitt [5].

Example. Let h_1, h_2, h_3 be 3 k-DNF formulas. Then $\text{MAJ}\{h_1, h_2, h_3\} = (h_1 \wedge h_2) \vee (h_2 \wedge h_3) \vee (h_3 \wedge h_1)$ which can be expanded into a $2k$-DNF formula. This is not good enough for Theorem 4, but it allows us to conclude that pac-learnability of k-DNF implies compression of k-DNF into $2k$-DNF.

3 Applications

We demonstrate how our KC-based Occam's razor theorem might be *conveniently* used, providing better sample complexity than the length-based version. In addition to giving better sample complexity, our new KC-based Occam's razor theorem, Theorem 1, is easy to use, as easy as the length based version, as demonstrated by the following two examples.

While it is easy to construct an artificial concept class with extremely bad representations such that our Theorem 1 gives *arbitrarily* better sample complexity than the length-based sample complexity given in Equation 5, we prefer to give real examples.

Application 1: Learning a String.

The DNA sequencing process can be modeled as the problem of learning a super-long string in the pac model [9, 10]. We are interested in learning a target string t of length say 3×10^9 (length of a human DNA sequence). At each step, we can obtain as an example a substring of this sequence of length n, from a random location of t (Sanger's Procedure). In current practice, $n \approx 500$, and sampling is very expensive. Formally, the concepts we are learning are sets of possible length n substrings of a superstring, and these are naturally represented by the superstrings. We assume a minimal target representation (which may not hold in practice). Suppose we obtain a sample of m substrings (all positive examples). In biological labs, a Greedy algorithm which repeatedly merges a pair of substrings with maximum overlap is routinely used. It is conjectured that Greedy produces a common superstring t' of length at most $2s$, where s is the optimal length (NP-hard to find). In

[2], we have shown that $s \leq |t'| \leq 4s$. Assume that $|t'| \approx 2s$.[1] Using the length-based Occam's razor theorem, this length of $2s$ would determine the sample complexity, as in Equation 6, with $p(n, s, \delta/2) = 2 \cdot 2s$ (the extra factor 2 is the 2-logarithm of the size of the alphabet $\{A, C, G, T\}$). Is this the best we can do? It is well-known that the sampling process in DNA sequencing is a very costly and slow process.

Let's now improve the sample complexity using our KC-based Occam's razor theorem.

Lemma 7 *Let t be the target string of length s and t' be the superstring returned by Greedy of length at most $2s$. Then*

$$K(t'|t, s, n) \leq 2s(2\log s + \log n)/n.$$

Proof. We will try to give t' a short description using some information from t. Let $S = \{s_1, \ldots, s_m\}$ be the set of m examples (substrings of t of length n). Align these substrings with the common superstring t', from left to right. Divide them into groups such that each group's leftmost string overlaps with every string in the group but does not overlap with the leftmost string of the previous group. Thus there are at most $2s/n$ such groups.

To specify t', we only need to specify these $2s/n$ groups. After we obtain the superstring for each group, we re-construct t' by optimally merge the superstrings of the neighboring groups. To specify each group, we only need to specify the first and the last string of the group and how they are merged. This is because every other string in the group is a substring of the string obtained by properly merging the first and last strings. Specifying the first and the last strings requires $2\log s$ bits of information to indicate their locations in t and we need another $\log n$ bits to indicate how they are merged. Thus $K(t'|t, n) \leq 2s(2\log s + \log n)/n$. □

This lemma shows that Equation 6 can also be applied with $p(n, s, \delta/2) = 2 \cdot 2s(2\log s + \log n)/n$, giving a factor $n/(2\log s + \log n)$ improvement in sample-complexity. Note that in DNA practice, we have $n = 500$ and $s = 3 \times 10^9$. The sample complexity is reduced over "length based" Occam's razor by a multiplicative factor of $n/(2\log s + \log n) \approx \frac{500}{2\times 31+9} \approx 7$.

Application 2: Learning a Monomial.

Consider boolean space of $\{0, 1\}^n$. There are two well-known algorithms for learning monomials. One is the standard algorithm.

Standard Algorithm.

1. Initial Concept: $m = x_1\overline{x_1} \ldots x_n\overline{x_n}$.

2. For each positive example, delete from m the variables that would make the example false.

[1] Although only the $4s$ upper bound was proved in [2], it is widely believed that $2s$ holds.

3. Return the resulting monomial m.

Haussler [7] proposed a more sophisticated algorithm based on set-cover approximation as follows.

Haussler's Algorithm.

1. Use only negative examples. For each literal x, define S_x to be the set of negative examples such that x falsifies these negative examples. The sets associated with the literals in the target monomial form a minimum set cover of negative examples.

2. Run the approximation algorithm of set cover, this will use at most $k \log m$ sets or, equivalently, literals in our approximating monomial. Here k is the number of variables in the target monomial.

It is commonly believed that Haussler's algorithm has better sample complexity than the standard algorithm. We demonstrate that the opposite is sometimes true (in fact for most cases), using our KC-based Occam's razor theorem, Theorem 1. Let's assume that our target monomial M is of length $n - \sqrt{n}$. Then the length-based Occam's razor theorem gives sample complexity n/ϵ for both algorithms, by Formula 6. However, $K(M'|M) \leq \log 3\sqrt{n} + O(1)$, where M' is the monomial returned by the standard algorithm. This is true since the standard algorithm always produces a monomial M' that contains *all* literals of the target monomial M. Also, we only need $\log 3\sqrt{n} + O(1)$ bits to specify whether other literals are in or not in M'. Thus our Equation 6 gives the sample complexity of $O(\sqrt{n}/\epsilon)$. In fact, as long as $|M| > n/\log n$ (which is most likely to be the case if every monomial has equal probability), it makes sense to use the standard algorithm.

4 Conclusions

Several new problems are suggested by this research. If we have an algorithm that, given a length-m sample of a concept in Euclidean space, produces a consistent hypothesis that can be described with only $m^\alpha, \alpha < 1$ symbols (including a symbol for every real number; we're using uncountable representation alphabet), then it seems intuitively appealing that this implies some form of learning. However, as Board and Pitt noted in their paper [5], the standard proof of Occam's Razor does not apply, since we cannot enumerate these representations. The main open question is under what conditions (specifically on the real number computation model) such an implication would nevertheless hold.

Can we replace the exception element or majority of 3 requirement by some weaker requirement? Or can we even eliminate such closure requirement and obtain a complete reverse of Occam's

razor theorem? Our current requirements do not even include things like k-DNF and some other reasonable classes.

5 Acknowledgements

We wish to thank Tao Jiang for many stimulating discussions.

References

[1] M. Anthony and N. Biggs, *Computational Learning Theory*, Cambridge University Press, 1992.

[2] A. Blum, T. Jiang, M. Li, J. Tromp, M. Yannakakis, Linear approximation of shortest common superstrings. *STOC'91.* To appear in *JACM.*

[3] A. Blumer and A. Ehrenfeucht and D. Haussler and M. Warmuth, Learnability and the Vapnik-Chervonenkis Dimension. *J. Assoc. Comput. Mach.*, 35(1989), 929-965.

[4] A. Blumer and A. Ehrenfeucht and D. Haussler and M. Warmuth, Occam's Razor. *Inform. Process. Lett.*, 24(1987), 377-380.

[5] R. Board and L. Pitt, On the necessity of Occam Algorithms. 1990 *STOC*, pp. 54-63.

[6] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant. A general lower bound on the number of examples needed for learning. *Inform. Computation*, 82(1989), 247-261.

[7] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36(2), pp. 177-222.

[8] D. Haussler, N. Littlestone, and, M. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. *29th FOCS*, 1988.

[9] T. Jiang and M. Li. DNA sequencing and Shortest Common Superstring. Submitted for publication.

[10] M. Li. Towards a DNA sequencing theory. *31st IEEE Symp. on Foundations of Comp. Sci.*, 125-134, 1990. (Abstract in *COLT'90.*)

[11] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 1993.

[12] L. G. Valiant. A Theory of the Learnable. *Comm. ACM*, 27(11), 1134-1142, 1984.

[13] M.K. Warmuth. Towards representation independence in PAC-learning. In *AII-89*, pp. 78-103, 1989.