COLT '88

PROCEEDINGS OF THE 1988 WORKSHOP ON COMPUTATIONAL LEARNING THEORY

Edited by David Haussler and Leonard Pitt

Proceedings of the 1988 Workshop on

Computational Learning Theory

MIT
August 3–5, 1988

David Haussler
(University of California, Santa Cruz)

Leonard Pitt
(University of Illinois)

Morgan Kaufmann Publishers
2929 Campus Drive
Suite 260
San Mateo, CA 94403

Senior Editor *Bruce Spatz*
Coordinating Editor *Beverly Kennon-Kelley*
Production Manager *Shirley Jowell*
Production Assistant *Elizabeth Myhr*
Cover Designer *Jo Jackson*
Compositor *Kennon-Kelley Graphic Design*

Library of Congress Cataloging-in-Publication Data

Workshop on Computational Learning Theory (1st : 1988 : Carnegie Mellon University)
Proceedings of the 1988 Workshop on Computational Learning Theory / [edited by] David Haussler.
p. cm.
First Workshop held in Carnegie Mellon University.
Bibliography: p.
Includes index.
1. Machine learning--Congresses. I. Pitt, Leonard. II. Title
0325.W67 1988
006.3'1--dc19 88-39431
CIP

This volume is based on papers presented at the First Workshop on Computational Learning Theory. While organizational expenses and some participant expenses were supported by grants from the Office of Naval Research and the National Science Foundation, the individual research papers appearing herein were not developed as a result of these grants. Consequently, the following statements may have limited applicability with respect to the contents of this volume.

This work relates to Department of Navy Grant N00014-88–J-1166 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

This material is based upon work supported by the National Science Foundation under Grant No. IRI 8815747. The Government has certain rights in this material.

ISBN 0-55869-019-5
Morgan Kaufmann Publishers, Inc.
2929 Campus Drive
San Mateo, CA 94403

93 92 91 90 89 5 4 3 2 1

Foreword

On three hot days in August 1988, about 70 people gathered in Boston to discuss new ideas and results on inductive inference, pattern recognition and machine learning. While a variety of disciplines were represented, the majority of those attending were theoretical computer scientists, primarily interested in the theory and analysis of learning algorithms. This volume is the official proceedings of that meeting, which was called the (First) Workshop on Computational Learning Theory. It contains papers for the talks that were presented in the technical sessions of the workshop, abstracts of some of the work in progress reported by the attendees, and a summary of the panel discussion "The Role of Theory in Learning Research."

Since, in light of the exchanges during and after the panel discussion, it is unlikely that any two people attending could agree on what computational learning theory is, no attempt will be made to define the term here. However, it is clear that the spirit of work can be traced back to the early work on inductive inference by Gold, Blum & Blum and others, as recently surveyed in [2]. Subsequent work in this area has covered a wide spectrum of issues, from recursion-theoretic characterizations of inferable function classes to the design of practical learning algorithms for particular cases. The papers of Case, Kurtz & Royer, Daley, Gasarch & Smith, Gasarch, et al., and Cherniavsky, et al. in this volume represent new contributions to the more abstract aspects of this line of investigation, while the papers of Sakakibara, Marron, Li & Vazirani, Ibarra & Jiang, and Porat & Feldman deal with more concrete questions concerning the problem of learning automata and formal languages.

One particular inference model that addresses the problem of designing practical learning algorithms is the learning framework recently proposed by Valiant [5]. This model combines polynomial restrictions on computational resources with a weaker, probabilistic criterion for successful inference. Applications of this framework to machine learning problems in artificial intelligence work are discussed in [3] and [4]. A more general viewpoint on this and related frameworks is given in [1]. In this volume, the papers of Haussler, et al., Linial, et al., Rivest & Sloan, Benedek & Itai, Vitter & Lin, Boucheron & Sallantin, Ehrenfeucht, et al., Shvaytser, Sloan, Shackleford & Volper, and Ehrenfeucht & Haussler give further results within this framework and close variants.

It is clear that much of the recent increased attention on computational learning in the scientific community as a whole is related to the sudden widespread interest in neural networks as models of computation. Several papers in this volume specifically address the problem of learning in artificial neural networks (Judd, Blum & Rivest, Raghavan, and Valiant), from the perspective of theoretical computer science. We believe that work on this problem may benefit from this perspective, and expect to see more participation from the theoretical computer science community in this area in the future.

The remaining papers discuss more abstract problems related to making predictions (Haussler, et al., Laird, and DeSantis, et al.) and learning Horn sentences (Angluin and Banerji). Many additional topics are addressed in the abstracts of work in progress. Some of these abstracts should also be consulted for further work on topics discussed in the technical papers. In particular, further results on topics discussed in the papers of Li & Vazirani,

Ibarra & Jiang, and Porat & Feldman are mentioned in the abstracts of Kearns & Valiant and Pitt & Warmuth. We note also that the question of efficiently learning regular languages by equivalence queries, mentioned as open in the Ibarra & Jiang paper, was resolved shortly after the workshop [6].

In short, this volume represents a rather eclectic group of papers from the emerging, interdisciplinary field of computational learning theory. It is our belief that this area of investigation will be fruitful in the coming years, and it is our hope that this workshop and the resulting proceedings will contribute to its development. We wish to thank all those who made this workshop possible, including the program committee (Dana Angluin, John Cherniavsky (chair), Ron Rivest, Carl Smith, Leslie Valiant, and Manfred Warmuth); local arrangements (Be Hubbard and Ron Rivest) and our sponsors at the National Science Foundation and the Office of Naval Research. As this was the first workshop we have organized, it was certainly a "learning" experience for us. We hope that, in a different way, it is for you too.

David Haussler and Lenny Pitt, co-chairs

References

[1] Angluin, D., "Queries and concept learning," *Machine Learning,* 2(2), 1988, pp. 319-342.

[2] Angluin, D., and C. Smith, "Inductive Inference: Theory and Methods," *ACM Comp. Surveys,* 15(3), 1983, pp. 237-270.

[3] Haussler, D., "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artificial Intelligence,* 36, 1988, pp. 177-221.

[4] Kearns, M., M. Li, L. Pitt, and L. Valiant, "Recent results on Boolean concept learning," *Proc. 4th Int. Workshop on Machine Learning,* Morgan Kaufmann, Los Altos, CA., 1987, pp. 337-352.

[5] Valiant, L.G., "A theory of the learnable," *Comm. ACM,* 27(11), 1984, pp. 1134-1142.

[6] Angluin, D., "Negative results for equivalence queries," Tech. Rep. YALEU/DCS/RR-648, Yale University, September, 1988.

Acknowledgements

The following articles are reprinted with permission of the authors and the publishers.

DeSantis, A., G. Markowsky and M. Wegman, "Learning Probabilistic Prediction Functions," © 1988 by the IEEE. This paper also appears in the *1988 Symposium on the Foundations of Computer Science.*

Haussler, D., M. Kearns, N. Littlestone, and M. K. Warmuth,"Predicting {0,1} Functions on Randomly Drawn Points," © 1988 by the IEEE. This paper also appears in the *1988 Symposium on the Foundations of Computer Science.*

Linial, N., Y. Mansour, and R. L. Rivest, "Results on Learnability and the Vapnik-Chervonenkis Dimension," © 1988 by the IEEE. This paper also appears in the *1988 Symposium on the Foundations of Computer Science.*

Ehrenfeucht, A., D. Haussler, M. Kearns and L. G. Valiant, "A General Lower Bound on the Number of Examples Needed for Learning," ©1989 Academic Press Incorporated. Reprinted with permission from the *Journal of Information and Computation.*

Ehrenfeucht, A. and D. Haussler, "Learning Decision Trees from Random Examples," ©1989 Academic Press Incorporated. Reprinted with permission from the *Journal of Information and Computation.*

Contents

Learning in the Limit—Selected Issues

Prediction

Learning Formal Languages and Automata

PANEL DISCUSSION REPORT

ABSTRACTS OF WORK IN PROGRESS

Neural Networks

Learning in Neural Networks

extended abstract

Stephen Judd
COINS dept., University of Massachusetts, Amherst, MA 01003, U.S.A.
judd@cs.umass.edu

Abstract

We formalize a notion of learning that characterizes the training of feed-forward networks. In the field of learning theory, it stands as new model specialized for the type of learning problems that arise in connectionist networks.

We prove the general problem and several sub-cases *NP*-complete and show one very similar sub-case to have a polynomial time algorithm. The broad class of sub-cases is formed by constraining the network architecture to have bounded depth and unbounded width.

1 Introduction

We formalize a notion of loading information into connectionist networks that characterizes the training of feed-forward neural networks. The formulation is similar to Valiant's [Val84] in that we ask what can be feasibly learned from examples and stored in a particular data structure. Our data structure is more particular than Valiant's since he requires only that the result be a 'sentence' in a language described by syntactic rules. In our form, the result must be a *particular* 'sentence' where the 'words' are not known in advance but their position and relationships are fully specified. This corresponds to the problem of finding retrieval functions for each node in a given network.

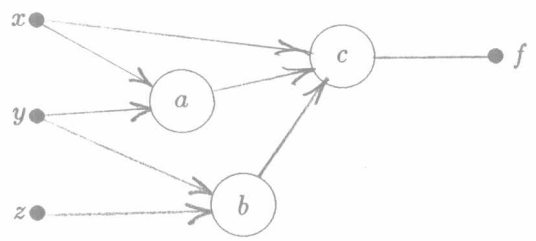
This loading problem is *NP*-complete for general networks, but we identify some polynomial-time problems yielded by placing constraints on the network architecture. The major focus of these constraints is on 'shallow' architectures which are defined to have bounded depth and unbounded width. We introduce a perspective on shallow networks, called the Support Cone Interaction (SCI) graph, which is helpful in distinguishing tractable from intractable subcases: When the SCI graph has tree-width $O(\log n)$, learning can be accomplished in polynomial time; when its tree-width is $n^{\Omega(1)}$ we find the problem *NP*-complete even if the SCI graph is a simple 2-dimensional planar grid. Tree-width is a generalization of the graph-theoretic notion of bandwidth.

2 Relationship to Valiant's Work

Valiant [Val84] established a definition of learning concepts from examples which has subsequently been elaborated by others.

Basically, Valiant's theory is intended to determine whether concepts *of a certain class* are feasibly learnable (i.e. requiring only polynomial time). For instance, if a concept can be expressed in conjunctive normal form with at most 4 variables per disjunct, is it possible to deduce that expression from seeing examples alone? The machine is required to find an expression in the given form that expresses whether a given bit string is a positive or negative example of a concept.

We [Jud87] initiated a different field of learning theory which is a direct formulation of a learning problem arising from the connectionist model of computation. That paradigm is concerned not just with what is feasibly learnable, but with what is feasibly learnable in a machine *with a certain fixed structure*. It requires that a representation for the learned data be found that can be embodied in a specific network structure. To achieve it, details of the function at each point in the net are alterable but no alterations to the connectivity of the network are allowed. For example, if the given network were this:



and the data to be learned were values for f, x, y, and z such that f were some function of x, y, and z, then the objective of the learning system would be not only to discover the function $f(x, y, z)$ but also to find 3 more functions a, b, and c such that $f(x, y, z) = c(x, a(x, y), b(y, z))$. This is more constrictive than Valiant's formulation because Valiant places only general grammatical guidelines on the form of f where we have an exact expression, minus only the specifications of a, b, and c. This prior knowledge of the form does not make the general learning paradigm any easier or harder, but merely different. It asks a question about whether a *particular* network can be made to represent some data, not whether it is possible to to find *some* network to represent those data.

3 A Formalization of Loading

3.1 The Learning Protocol

The type of problem investigated here is known as supervised learning. In this paradigm input patterns (called *stimuli*) are presented to a machine paired with their desired output patterns (called *responses*). The object of the learning machine is to remember all the associations presented during a training phase so that in a later testing phase the machine is able to emit the associated response for any given stimulus.

In what follows, every stimulus σ is a fixed-length string of s bits, and every response ρ is a string of r bits with "don't cares", that is $\sigma \in \{0,1\}^s$ and $\rho \in \{0,1,*\}^r$. The output from a net is an element of $\{0,1\}^r$. The purpose of a response string is to specify constraints on what a particular output can be: we say that an output string, θ, *agrees* with a response string, ρ, if each bit, θ_i, of the output equals the corresponding bit, ρ_i, of the response whenever $\rho_i \in \{0,1\}$. Each stimulus/response pair is called an *item*. A *task* is a set of items that the machine is required to learn and typically it has far fewer than 2^s items. To be reasonable every stimulus in a task should be associated with no more than one response, so we can think of a task as a partial function.

3.2 Network Architecture

The particular style of connectionist machines considered here is that of non-recurrent, or feed-forward, networks of computing elements. This is a generalized combinational circuit; the connections between nodes form a directed acyclic graph, and the nodes perform some function of their inputs as calculated by previous nodes in the graph.

We define an *architecture* as a 5-tuple $A = (P, V, S, R, E)$, where
P is a set of *posts*,
V is a set of n *nodes:* $V = \{v_1, v_2, \ldots, v_n\} \subseteq P$,
S is a set of s *input posts:* $S = P - V$,
R is a set of r *output nodes:* $R \subseteq V$, and
E is a set of directed *edges:* $E \subseteq \{(v_i, v_j) : v_i \in P,\ v_j \in V,\ i < j\}$
The constraints on the edges ensure that no cycles occur in the graph. Denote the set of *input posts* to node v_k as

$$pre(v_k) = \{v_j : (v_j, v_k) \in E\}.$$

3.3 Node Functions

Each node in a network contributes to the overall computation by taking signals from its input edges and computing an output signal. In our analysis, we consider only binary-valued functions.

$$f_i : \{0,1\}^{|pre(v_i)|} \rightarrow \{0,1\}$$

The function f_i is a member of a fixed set $\mathcal{F}$ of functions. For our purposes it is not too critical what the set is but our results hold for the customary case where $\mathcal{F}$ is the set of linear threshold functions.

A *configuration* of a network is an assignment of some node function to each node in the architecture:

$$F : V \rightarrow \mathcal{F},$$

where $f_i = F(v_i)$ means that f_i is the function that node i computes.

3.4 The Computational Problem

In a configured network, every node performs a particular function, and the network as a whole performs a particular composite function. An architecture, A, and a configuration, F, together define a (total) mapping from the space of stimuli to the space of responses

$$M_F^A : \{0,1\}^s \rightarrow \{0,1\}^r$$

and this defines the retrieval behaviour of a network.

Recall that an item in a task is a pair of strings (σ, ρ). When the posts in S are given the values of respective elements of σ, the network mapping defines values for each post in R. It is required that these retrieved values agree with respective elements of ρ.

The *loading* problem can now be defined.

Instance: An architecture A and a task T.

Question: Find a configuration F for A
such that $T \subseteq \{(\sigma, \rho) : M_F^A(\sigma)$ agrees with $\rho\}$.

This paper is a report on the computational complexity of the above decision problem and into some special cases of the problem.

4 Complexity Results

Theorem 1 *Loading is NP-complete even when*

- *the architectures are restricted to be of depth* ≤ 2 *and of fan-in* ≤ 3,
- *tasks are restricted to be monotonic,*
- *there are there are only two bits in the stimulus strings* ($s = |S| = 2$),
- *tasks are restricted to be of no more than 3 items,*
- *tasks are restricted to be performable by the network using only the node functions* AND *and* OR, *but* any *Boolean node functions may be used to find a configuration for it, and*
- *only 67% of the items are required to be retrieved correctly.*

□

A proof is available in [Jud88b] and different one can be found in [Jud87].

Definition In an architecture $A = (P, V, S, R, E)$, each output node $x \in R$ has a *support cone*, $\mathrm{sc}(x)$, which is the set of all nodes in V that can potentially affect the output of that node; that is, it is the set of predecessor nodes

$$\mathrm{sc}(x) = \{x\} \cup \{\mathrm{sc}(y) : y \in pre(x) \cap V\}$$

The network retrieval behaviour at any particular output node is determined by (and only by) the functions assigned to each node in its support cone.

Definition A *support cone interaction graph* (SCI graph) for an architecture, is an accounting of the interactions between support cones. It is a graph with nodes $\{z_1, z_2, \ldots, z_r\}$ corresponding one-to-one with the output nodes, R, and having edges $\{(z_i, z_j) : \mathrm{sc}(R_i) \cap \mathrm{sc}(R_j) \neq \emptyset\}$.

Definition A family of architectures is said to be *shallow* if the size of the largest support cone in each architecture is bounded by a polynomial in n, the number of nodes in the architecture.

The complete configuration space for any support cone in any architecture in a shallow family can be exhaustively searched in polynomial time.

Theorem 2 *Loading shallow architectures is NP-complete even if the SCI graph is a regular rectangular or hexagonal grid.* □

Definition The *tree-width* of a graph is defined by [RS86] in the following way: Let G be a graph with vertex set $V(G)$. A *tree-decomposition* of G is a family $\{X_i : i \in I\}$ of subsets of $V(G)$, together with a tree T with $V(T) = I$, which have the following properties:

- $\bigcup\{X_i : i \in I\} = V(G)$
- Every edge of G has both its ends in some $X_i (i \in I)$.
- For $i, j, k \in I$, if j lies on the path in T from i to j then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree-decomposition is $\max\{|X_i| - 1 : i \in I\}$. The *tree-width* of G is the minimum width over all possible tree-decompositions.

Graphs with tree-width $\leq k$ are also said to be embeddable in partial k-trees. This alternative (and earlier) definition appears in [ACP84]. The tree-width of a graph is never greater than its bandwidth.

Theorem 3 *Loading shallow architectures whose SCI graphs are of limited tree-width can be accomplished in polynomial time, provided that a tree-decomposition is given that exhibits the required width.* □

The theorem holds when the word 'limited' means $O(\log n)$ where n is the size of the architecture (not the size of the task).

Theorem 4 *For shallow architectures where the tree-width of their SCI graphs has a growth function of* $G(n) = n^{\Omega(1)} = n^{\epsilon}$, *loading is NP-complete.*

Proofs to theorems 3 and 4 appear in [Jud88a].

5 Conclusions

The problem of simply remembering a list of stimulus/response pairs is trivial on a Turing machine or a random access machine, but it has been shown to be very difficult when required to be put into a specific immutable circuit form. This has immediate ramifications for learning in connectionist networks, which typically depend on a form of on-line training by seeing example stimulus/response pairs. A theory of constraints on the loading problem leading to tractable subcases will translate into network design constraints and/or a description of what tasks they can learn.