7N-63-TM
91484
P-18

(NASA-TM-107858) EFFICIENT UNSUPERVISED LEARNING (NASA) 18 p
N92-70671
Unclas
Z9/63 0091484

Efficient Unsupervised Learning

PHILIP LAIRD

AI RESEARCH BRANCH, MAIL STOP 244-17
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-88-03-15-0

March, 1988

Efficient Unsupervised Learning

Philip D. Laird
NASA Ames Research Center
Moffett Field, CA 94035*

Abstract

The efficiency of learning from unclassified data (unsupervised learning) is examined by constructing a framework similar in style to the recent work on supervised concept learning inspired by Valiant. We define the framework and illustrate it with results on three model classes. The framework is compared to both the supervised learnability model and other models of unsupervised learning.

1 Introduction

Supervised concept learning occurs when a teacher presents the learner with data labeled according to whether the items are examples or counterexamples of the target concept. The learner must then solve a search problem in a pre-designated concept class. Unsupervised learning omits the teacher, and as a result the nature of the problem changes in an essential way. In this work we ask what it means for a hypothesis class to be learnable in an unsupervised setting, and if it is learnable, what it means to be efficiently learnable.

This research originated with ongoing efforts to construct efficient empirical learning programs applicable to NASA domains. Unsupervised learning problems tend to be more applicable than supervised concept-learning problems; most of the existing theory, however, has been developed for the latter. Even in the supervised model, however, the assumption of independent random examples is typically too strong; and the introduction of noise into the example sequence requires the addition of new nuisance parameters to the models. Existing theories of unsupervised learning devloped by statisticians, while mathematically elegant, are hard to analyze for computational complexity because they assume a numerical-analysis model of computation. By contrast, the model suggested here

- provides a computational theory for learning probability distributions in an unsupervised setting and for analyzing the learning complexity associated with the representation of a family of distributions.
- accounts for noise and temporal dependence among the examples, and suggests ways to extend this ability naturally to the supervised case.

The model has been chosen with experimental objectives in mind. The mathematical results are more exploratory than fundamental, but are sufficient to demonstrate the potential utility of the framework. Additionally, many interesting, unexplored problems present themselves for investigation.

*Internet: LAIRD@PLUTO.ARC.NASA.GOV.

2 Unsupervised Learning: brief overview

Many approaches to unsupervised learning have been studied over the years; of these, the most closely related to this work are the studies of finite-mixture distributions and the theory of hidden Markov models.

We can imagine a population consisting of a mixture of N component populations over the same vector space X. With the i'th component distribution f_i, we associate a probability p_i. f_i is assumed to be parameterized by some finite set of values θ_i (e.g., mean and standard deviation), which we indicate by writing $f_i(\mathbf{x} \mid \theta_i)$ ($\mathbf{x} \in X$). The mixture distribution f is then given by

$$f(\mathbf{x}) = \sum_{i=1}^{N} p_i f_i(\mathbf{x} \mid \theta_i).$$

Intuitively, the process described by such a distribution is one in which a population is selected with probability p_i, and then a vector is drawn from that population according to the probability distribution f_i. The learning problem is to identify N, the p_i, and θ_i from independent random samples of the population distributed according to f. Note that the distributions may overlap – i.e., the same vector may be selected in different ways if it is part of two or more populations.

Theoretical problems that have been studied in connection with mixture models are identifiability (the question of whether every mixture has a uniquely distinguishable set of components), sampling requirements (number of data points required for an acceptably close solution), and effective algorithms for computing the parameters from consistent estimators. Good references for this work are [6] and [7].

Markov modeling, subject to a variety of assumptions and conditions, has been applied to problems ranging from speech recognition to cell biology under the generic label of "hidden Markov models". Such a model consists of a finite Markov chain with, say, n states, and a set of n probability density functions f_s, one for each state in the chain. From its current state, the Markov process makes a transition to some state s, but the actual state of the process is not observable. Instead, a symbol $\mathbf{x} \in X$ is selected according to the distribution f_s and presented to the learner, whose task is to infer the underlying chain and associated distributions.

Baum and Petrie [3] demonstrate a maximum-likelihood convergence property for chains with a given number of states. Later, in the course of trying to apply this result, Baum and others [4] developed the ***re-estimation algorithm*** for determining the maximum-likelihood parameters of a given chain.

A different, but related, class of Markov models is obtained by associating with each transition of a Markov chain a symbol $\mathbf{x}$ from a finite set X. Thus from a given state s, the process selects a transition from s to another state s', according to the probability $p_{s,s'}$ assigned to that transition, and emits the symbol associated with that transition. The learning problem, again, is to infer the structure and parameters of the chain responsible for the observed sequence of symbols. Rudich [15] studied this problem using a maximum-likelihood strategy, with the goal of identifying the minimum-size chain in the limit from an infinite sample, assuming that the transition probabilities are rational. He considers first the case where an upper limit is provided on the number of states of the target chain, and later the case where no such limit is known.

In none of the above work, however, is the computational complexity of finding or using the models considered. Also, in view of the fact that in actual applications statistically dependent sample data is the norm, it is remarkable that learning from other than independent data has

been so consistently avoided in the literature.

3 Unsupervised Learning Models

Let us envision the learner as receiving a stream of input data, either as a continuing sequence of characters (e.g., bits) or as blocks of data, such as vectors or strings. In both cases, the data are discrete, both in structure and in time. Initially the learner perceives the input as random, since he has no prior basis for expectation or prediction. In time, however, patterns are discerned and regularities classified, until the learner is less surprised by what he sees. Ideally such a learner reaches the point where s/he can predict the next input unit with 100% accuracy. When randomness is inherent in the data, however, this ideal is unattainable without omniscience.

In this scenario, the learner infers statistical regularities in the data stream without the benefit of a teacher. We are viewing such ***unsupervised learning*** as the inference of a probability distribution for the next input unit. This definition turns out to be flexible enough to cover situations where the particular temporal sequence of input units is not of concern, as well as those in which the temporal aspects are significant. If an application entails making choices or decisions based on the predicted input, then utility theory can be applied to devise an optimal strategy, but we shall not pursue the learning problem beyond that of determining the probabilities.

In developing a formal model of unsupervised learning, we need to understand just what it is that we are learning, how the object we are trying to learn is presented, and what the criteria are for learning – specifically, when a class is learnable, and when it is efficiently learnable.

3.1 Model Classes

Recall that unsupervised learning entails extracting predictable aspects from an apparently random input stream. We call the representation of our predictions a ***model.***

Let the input stream be $x_1, x_2, \ldots, x_t, \ldots$ taking values in the finite set X. A *model* M is a procedure that, after receiving as input $x_1, \ldots, x_{t-1}$ but before receiving x_t, is capable of computing a (conditional) probability distribution $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$ of the next $x_t \in X$. Thus, before seeing x_t, M is able to compute for any $x \in X$ a rational fraction in $[0,1]$, which we interpret as its prediction for the conditional probability $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$. To say that M computes a probability distribution means that the values of $\mathbf{P}$ over all x are normalized such that $\sum_{x \in X} \mathbf{P}(x \mid x_1 \ldots x_{t-1}) = 1$. After seeing x_t, M updates its "state" to reflect the value x_t, so that it can then compute $\mathbf{P}(x_{t+1} \mid x_1 \ldots x_t)$. This process continues forever.

In addition to these functional requirements, we impose the following efficiency requirements: the time for M to compute any prediction $\mathbf{P}(x_t \mid x_1 \ldots x_{t-1})$ and the time for it to update its state to account for x_t are each bounded as a function of t. Intuitively, this means that M doesn't "slow down" as time goes on; it is able to keep up with the incoming data stream and remain an on-line predictor.

The formal definitions are given below, along with a running example. Let X be a fixed[1] set of symbols, and let $Q[0,1]$ be the set of rationals over the interval $[0,1]$.

[1] and in this paper, finite

Definition 3.1 *A* well-parameterized family of distributions *is a finite set* Λ *of parameter values, with an associated algorithm* $\mathbf{P}_\Lambda$ *that computes a probability distribution on* X *for each* $\lambda \in \Lambda$*:* $\mathbf{P}_\Lambda(x,\lambda) \in Q[0,1]$*, and* $\sum_x \mathbf{P}_\Lambda(x,\lambda) = 1$*, for all* λ.[2]

Definition 3.2 *Fix a well-parameterized distribution family* Λ*. A* model M *is an algorithm that receives as input an infinite stream* $x_1, x_2 \ldots \in X^\infty$ *and in turn outputs a stream* $\lambda_1, \lambda_2 \ldots \in \Lambda^\infty$ *of parameters from* Λ*.* M *outputs* λ_t *after receiving* x_{t-1} *but before receiving* x_t*; we interpret* $\mathbf{P}_\Lambda(\cdot, \lambda_t)$ *as* M*'s estimate for the conditional probability distribution* $\mathbf{P}(\cdot \mid x_1, \ldots, x_t)$*. Furthermore, the time for* M *to compute and output* λ_t *must be bounded (as a function of* t*).*

Intuitively the series of parameters output by a model represent the model's vision of how the probability distribution over the input varies with time. If the model views the world as static (successive x_i's are independent), then it outputs the same parameter in each round. Note that we do *not* view a model as a learning process: it is the job of a separate *learner* to find the right model.

A set $\mathcal{M}$ of models over the same X and Λ is referred to as a *model class*. The model class is to unsupervised learning what the concept class is to supervised learning.

Example 3.3 Suppose $X = \{0,1\}$. Let Λ be a finite subset of $Q[0,1]$, and for each $\lambda \in \Lambda$, let $\mathbf{P}(x,\lambda) := (\text{ if } x = 1 \text{ then } \lambda \text{ else } 1-\lambda)$. This describes a family of Bernoulli distributions (biased coin flips) where the probability is λ of coming up with a "1". (By contrast, the family of *all* Bernoulli distributions, for which $\Lambda = Q[0,1]$, is not well parameterized because Λ is infinite.)
Fix $\lambda \in \Lambda$, and consider the model M_λ that outputs λ for every t. This model corresponds to a view of the world in which the environment is conducting a sequence of independent coin flips, yielding 1's and 0's with probability λ and $1-\lambda$, respectively. The class $\mathcal{M}$ of models of this type, one for each parameter in Λ, represents a choice of world-views that agree about the nature of the environment (coin-flipping) but disagree about the probabilities. The "real" world may not be any of these, but over a given period of time, one of these models is likely to be the best of its class in accounting for the frequency of 1's and 0's. $\triangle$

When a growth parameter n is given, we have a *model family* $\mathcal{M}_n$, associated with X_n and Λ_n ($n \geq 1$), provided: (*i*) The algorithm $\mathbf{P}_n$ (corresponding to Λ_n) computes the probability $\mathbf{P}_n(x,\lambda)$ in time bounded by a polynomial in n; and (*ii*) each model M in $\mathcal{M}_n$ computes its output λ_t in time uniformly bounded by a polynomial in n. This means that our model families must be able to scale upward at a reasonable rate to handle larger sized models and possibly a larger set of input symbols.

Example 3.4 (Bernoulli family.) For $n \geq 1$, let Λ_n be the subset of rationals λ in $Q[0,1]$ whose length $\text{len}(\lambda)$, when written in binary, is at most n bits. Let $\mathcal{M}_n$ be the class of models corresponding to Λ_n, as in Example 3.3. Then $X = \{0,1\}$, Λ_n, and $\mathcal{M}_n$ together define a family of Bernoulli models. The time for a model $M \in \mathcal{M}_n$ to output its prediction is $\mathcal{O}(n)$, since all it has to do is output up to n bits. For the same reason, $\mathbf{P}_n$ requires $\mathcal{O}(n)$ time to compute probability estimates. $\triangle$

[2] When X is infinite, the algorithm $\mathbf{P}_\Lambda$ needs to be efficient as a function of the length of x.

3.2 Presentation

In this work we assume only that the input is a stream $\{x_t,\ t \geq 1\} \in X^\infty$ of symbols from a finite set X. It is convenient to regard the symbols as arriving one after another at a steady rate. In practice, this rate strongly affects how large an n we can support for our model class $\mathcal{M}_n$ on a given machine.

3.3 Learnability

Having selected a model class $\mathcal{M}$, we then have a learning problem: to find a model in $\mathcal{M}$ that is "optimal" (in some sense) for predicting the input data stream.

How do we decide which model is best? In general there may be no "best" model. But when in fact the input *really is* the result of a model M in the class, we should certainly expect, disregarding the computational resources required, that some algorithm eventually could identify M, or an equivalent, as the best model.

Definition 3.5 *Two models M and M' with outputs in Λ are said to be* equivalent *if, for any input stream, the values of $\mathbf{P}(x, \lambda_t)$ and $\mathbf{P}(x, \lambda'_t)$ are the same for all t and x – i.e., the probability distributions predicted by the two models are always identical, even when their outputs λ_t and λ'_t are different.*

Definition 3.6 *A model class $\mathcal{M}$ is* learnable in the limit *if there exists a procedure $\mathcal{A}$ with the following properties:*

1. *In response to the stream of inputs $\{x_t,\ t \geq 1\} \in X^\infty$, $\mathcal{A}$ outputs a stream of models $\{M_t,\ t \geq 1\}$ in $\mathcal{M}^\infty$. It is convenient, but not necessary, to consider M_t as the response of $\mathcal{A}$ to the input x_{t-1}.*
2. *Let M be any model in $\mathcal{M}$, and let the input stream be generated according to the probability distribution determined by M. Then with probability one, all but finitely many of the models M_t output by $\mathcal{A}$ are equivalent to M.*

A model that is "way off" in describing the input will over time predict very small probabilities for the inputs that actually occur. By contrast, a better model finds that its predictions come true more often, in that the probabilities for observed inputs are higher. The *likelihood* is a well-known measure of the quality of predictions over the history of the process.

Definition 3.7 *Let M be a model and $X^t \equiv x_1 \ldots x_t$ a sequence of t inputs. Let $\lambda_1, \ldots, \lambda_t$ be the parameter outputs of M over this sequence. The* likelihood *of X^t according to M is given by the product*

$$L(X^t \mid M) = \mathbf{P}(x_1, \lambda_1) \ldots \mathbf{P}(x_t, \lambda_t).$$

The likelihood function is useful in proving learnability in the limit. The idea behind this type of proof is as follows. Let the input be generated by $M \in \mathcal{M}$, and let M' be any model in $\mathcal{M}$. One shows that with probability one,

$$L(X^t \mid M) \geq L(X^t \mid M') \tag{1}$$

for all but finitely many t. Furthermore, equality holds iff M and M' are equivalent. Thus a learning algorithm that simply outputs the model with the maximum likelihood for the input so far satisfies the learnability requirement.

The following result, for example, is obtained from this type of approach. Various proofs are available in the statistics literature; see [13] for a simple proof.

Proposition 3.8 *Let X be a finite set and $\mathcal{M}$ a finite class of models predicting statistically independent inputs over X. Then $\mathcal{M}$ is learnable in the limit.*

As an immediate corollary, Bernoulli models are learnable in the limit. In interesting question is for what processes Eq. (1) is also a *necessary* condition for learnability. Also, in many situations finding a model is not enough: we want to find a *minimum-size* model equivalent to the source. Results of this kind are obtained in [13].

3.4 Efficient Learnability

Learnability in the limit does not necessarily imply uniform convergence over time – i.e., gradually improving the quality of the model. But while infinite convergence results are not especially useful by themselves, they frequently point the way toward finite algorithms: run the infinite algorithm for a certain period of time, stop it, and see what you've got.

We note that (as with supervised learning) a finite algorithm can usually be expected only to approximate the input source. How do we measure the closeness of such an approximation?

Definition 3.9 *Let $\mathcal{M}$ be a model class. A* distance *function* d *is a non-negative real function on pairs of models $\mathcal{M} \times \mathcal{M}$ such that: (i) for equivalent models, M and $M' \in \mathcal{M}$, $\mathrm{d}(M' \mid M) = 0$; (ii) for inequivalent models, $\mathrm{d}(M' \mid M) > 0$.*

Example 3.10 Let M be the target model, and M' any other model. A rather strong distance function is the maximum possible difference in the predicted probabilities of the two models:

$$\mathrm{d}_1(M' \mid M) = \max_{x_t} |\mathrm{P}(x_t, \lambda'_t) - \mathrm{P}(x_t, \lambda_t)|.$$

Another popular distance function is variously known as the cross entropy, informational divergence, or Kullback-Leibler function [11, 12, 16]:

$$\mathrm{d}_2(M' \mid M) = \lim_{t \to \infty} \frac{1}{t} \log \frac{L(X^t \mid M)}{L(X^t \mid M')},$$

assuming the limit exists. For ergodic processes with the likelihood property (Eq. 1), this is a well-defined distance function. The fact that this function is not symmetric makes it somewhat difficult to work with.

The mean-squared error, most often used for time-independent processes, is defined by

$$\mathrm{d}_3(M' \mid M) = \lim_{t \to \infty} \sum_{i=1}^{t} \frac{[\mathrm{P}(x_i \mid \lambda') - \mathrm{P}(x_i \mid \lambda)]^2}{t},$$

again assuming the limit exists. For processes consisting of independent random events, this limit is the expected value of the squared difference in predicted probabilities. $\triangle$

Given such a distance function d, we divide the efficiency question into two parts. Let $\mathcal{M}_n$ be a (learnable) model family, with n as the growth parameter.

Definition 3.11 *$\mathcal{M}_n$ is said to be* data efficient *(with respect to the distance function* d*) if there exists a polynomial f and an algorithm $\mathcal{A}$, taking as parameters n, ϵ, and δ, with the following property: for an input stream generated by any $M \in \mathcal{M}$, $\mathcal{A}$ examines $t \leq f(n, 1/\epsilon, 1/\delta)$ inputs $x_1 \ldots x_t$ and outputs a model $M' \in \mathcal{M}_n$ such that $\mathrm{d}(M' \mid M) \leq \epsilon$, with probability $\geq 1 - \delta$.*

Below, we give examples of model families that are and are not data efficient, with respect to different distance functions.

As with supervised learnability, there are families that we know are data efficient but for which we do not yet have algorithms to find good models from the input data. The ability to find good models quickly from a sufficient sample of the environment is captured by the following definition.

Definition 3.12 *$\mathcal{M}_n$ is said to be* efficiently learnable *if it is data efficient, and the learning algorithm $\mathcal{A}$ runs in time bounded by a polynomial in the total size of the input data.*

The efficiencies of learning several sample model families have been investigated; some results are given below (see [13] for the complete analysis). To complete our running example, we consider here the Bernoulli family – probably the simplest non-trivial model family.

Example 3.13 (Bernoulli family.) The Bernoulli family of models was defined in Example 3.4. There are 2^n possible models in this family; the fact that we can efficiently find an ϵ-approximate model is, of course, a direct consequence of the exponential convergence of sums of independent random variables to their mean; and the results below generalize rather easily to multinomial and related stochastic processes, suitably parameterized.

Assume the environment is a sequence of 1's and 0's generated by a Bernoulli process whose parameter is $\lambda_* \in \Lambda_n$. Let d_1 be our distance function. Given ϵ and δ, we want to choose $\lambda \in \Lambda_n$ such that $|\lambda - \lambda_*| \leq \epsilon$, with probability $\geq 1 - \delta$.

Define ϵ' to be $2^{-n}/3$ if $\epsilon < 2^{-n}$, or $\lfloor \epsilon \cdot 2^n \rfloor \cdot 2^{-n}$ otherwise. We argue that at most

$$t = \left\lceil \frac{\log_e(2/\delta)}{2\epsilon'^2} \right\rceil \tag{2}$$

inputs are required to obtain an ϵ-approximate parameter λ. (Note that the sample size t is clearly polynomial in n, $1/\epsilon$, and $\log 1/\delta$.) Given this many inputs, let t_1 be the number of 1's in the string X^t. Our algorithm is to determine the proportion t_1/t of 1's and to round this estimate to the nearest multiple of 2^{-n}; this value is output as our hypothesis λ.

To see that t is sufficiently large, consider first the case where $\epsilon \geq 2^{-n}$. Thus $\epsilon' \leq \epsilon$, and by Hoeffding's Inequality [10], in t trials,

$$\begin{aligned} \Pr\left[\left|\frac{t_1}{t} - \lambda_*\right| \geq \epsilon'\right] &\leq 2e^{-2t\epsilon'^2} \\ &\leq \delta. \end{aligned}$$

Thus with sufficiently high probability, the (rational) estimate t_1/t is within ϵ' of λ_*. Consider now the effect of rounding. ϵ' is ϵ reduced to the next lower multiple of 2^{-n}. If $\lambda_* = i2^{-n}$ and $\epsilon' = j2^{-n}$ for some integers i and j, then since with high probability t_1/t is within ϵ' of λ_*,

$$(i - j)2^{-n} \leq t_1/t \leq (i + j)2^{-n}.$$

Rounding t_1/t gives λ, for which we have

$$(i-j)2^{-n} \leq \lambda \leq (i+j)2^{-n},$$

and so $|\lambda - \lambda_*| \leq \epsilon' \leq \epsilon$.

In the other case, when $\epsilon < 2^{-n}$, ϵ' is fixed at $\frac{1}{3}2^{-n}$. Hoeffding's Inequality still guarantees that t_1/t will be within ϵ' with high probability, and rounding will therefore result in $\lambda = \lambda_*$. We have thus proved:

Theorem 3.14 *The family of Bernoulli models is data efficient, with respect to the* $\mathbf{d}_1$ *distance. Furthermore, when* $\epsilon < 2^{-n}$*, the models are learnable exactly from polynomially many input data, with high probability* ($> 1-\delta$).

As a corollary, we note that the above algorithm for processing the t data values runs in time polynomial in t. Hence *the family of Bernoulli models is efficiently learnable (with respect to* $\mathbf{d}_1$*).*

Suppose we use a different distance function? Obviously, the above results also hold for the $\mathbf{d}_3$ distance function. It can be shown that Bernoulli models are still data efficient and efficiently learnable when the $\mathbf{d}_2$ distance function is used, but the algorithm changes slightly to avoid singularities in the $\mathbf{d}_2$ function, and our upper bound on the size t of the data sample differs from that given above for $\mathbf{d}_1$ by a small constant factor.

Suppose next that our model class does not correspond to reality, that the actual probability λ_* is not a multiple of 2^{-n}. Then the algorithm above will still produce a parameter $\lambda \in \Lambda_n$, but as it happens, λ may not be within ϵ of λ_* because rounding could push it off in the wrong direction. A simple change to the calculation of ϵ' will fix this small problem, but this illustrates that, even if an algorithm correctly finds an ϵ-approximation to any model within the hypothesized model class $\mathcal{M}_n$, that same algorithm may not produce an ϵ-approximation when the data is *not* a result of any model in the class, and may not even find the model in the class that most closely describes the data. $\triangle$

4 A Classification Model Family

Let r be a positive integer and X be the set of ordered n-tuples (vectors) $(x_1, \ldots, x_n)$, with each element ("attribute") x_i taking integer values in the set $\{0, \ldots, r-1\}$. Thus the cardinality of X is r^n. Suppose each vector in X is assigned a fixed probability, and that at regular intervals a vector is selected independently and at random from this distribution and presented. In such a case the optimal model may require at least r^n bits to write down the distribution – in violation of the polynomial scaling requirement for a model family. Efficiency thus dictates that vectors be grouped into large sets, and that from the parameters of a set we be able to compute the probability of any vector in the set. This is the motivation for our classification model.

We define a very simple family of classification models, but one that nevertheless presents interesting problems for learning. A model partitions the vector space into a non-empty finite set of disjoint orthogonal hyperboxes. The boxes are defined by a constraints of the form $i \leq x_i \leq j$ for each attribute ($0 \leq i, j \leq r$). In two dimensions, for example, the $r \times r$ square X is partitioned into rectangles. Associated with each box S_i of the partition is a probability π_i, interpreted as the probability that one of the vectors in set S_i will occur next. If the vector

$\mathbf{x}$ is in set S_i, then its probability $\mathbf{P}(\mathbf{x})$ of occurring next is understood to be $\pi_i/|S_i|$. Thus specifying a partition and a normalized set of probabilities for each block has the effect of assigning a probability to each vector in the space.

Initially we shall limit the lengths (in bits) of the set probabilities π_i to k binary bits. Naturally we require that the probabilities of the boxes sum to 1. Furthermore the number of boxes in the partition will be bounded, and for simplicity, we adopt the same constant k for this bound.

A classification model can be encoded in about $nk \log r$ bits by specifying the constraints and probabilities of each of the rectangular sets of the partition. Λ_n denotes the set of all such valid specifications. A classification model M_λ outputs the encoding λ of a model before each input value x_t is received. $\mathcal{M}_n$ denotes the family of all such models, as a function of n; we refer to this family as k-RDC (bounded rectangular disjoint classifications). It is easy to verify that Λ_n is well-parameterized by designing an $\mathcal{O}(n)$ algorithm to compute the probabilities of a vector $\mathbf{x}$ from a model parameter λ.

The view of the world painted by these assumptions is that of an environment that selects, independently for each time t, one of a small number of classes, and chooses at random a vector from that class for presentation. Note that, unlike the finite-mixture model discussed above, no vector belongs to more than one class. As with the coin-flip model, the actual environment may not be generated in this fashion, but the best model in the class is likely to capture some aspects of the environment, by classifying adjacent vectors into a few (hopefully meaningful) classes.

k-RDC is learnable in the limit, by Proposition 3.8; but we are interested primarily in the efficiency of learning. Suppose a stream of vectors is being generated by a k-RDC model, and consider how we might go about identifying the source model. Proposition 3.8 and identification-in-the-limit give us a generous hint: eventually a correct model will have a higher likelihood than any incorrect one. The question is, how large a sample is needed before all "bad" models – i.e., ones that are not ϵ-approximations to the source – have significantly lower likelihood, with high probability?

Let M_* be the source k-RDC model, and let M_1 be a k-RDC model based on the partition $\mathcal{R} = \{R_i\}$ and assigning a probability of π_i' to the block R_i. Assume we have a sample X^t of t vectors, randomly generated according to the statistics determined by M_*. In studying the learnability of this model class, we need two technical preliminaries (details are given in [13]).

Lemma 4.1 *Fix any $\delta > 0$. Let R be a box in any k-RDC partition (not necessarily the target model M_*). The probability π_R that the next input vector belongs to R can be determined exactly, with probability more than $1 - \delta$, from a sample of t vectors, where t is bounded by a polynomial in n and* $\log(1/\delta)$.

The proof appeals to the previous result for the Bernoulli family. The main observation is that any box R in a k-RDC partition contains a large number of vectors, at least r^{n-k+1} in fact. (This is a consequence of requiring that the boxes "tile" the vector space.) One then argues that the total probability π_R for the vectors in R is either zero or an integral multiple of a constant β depending only on k. Hence learning π_R by sampling is a matter of learning a Bernoulli parameter, which we can do using a sample of size $\mathcal{O}(\log 1/\delta)$.

Lemma 4.2 *For any $\delta > 0$, a sample of $\mathcal{O}(\log(n/\delta))$ vectors suffices to determine exactly the maximum-likelihood parameters (the π_R's) of all k-RDC partitions, with probability $> 1 - \delta$.*

For, if there are N admissible boxes, and if the maximum-likelihood parameters (the π's) of each are learned exactly with probability $1-\delta/N$, then the probability that any one parameter is learned incorrectly is less than δ. By Eq. 2, with $\epsilon' \equiv \beta/3$, our sample need be no larger than

$$t = \left\lceil \frac{\log_e(2N/\delta)}{2\epsilon'^2} \right\rceil.$$

And N can be bounded above by $(2nr)^k$.

Theorem 4.3 *The family k-RDC is data efficient with respect to any distance function.*

PROOF: The preceding lemmas show that we can determine the actual probabilities π_R for all the blocks of all possible partitions using a data-efficient sample of vectors. We now consider how to use these values to determine a model equivalent to the target M_*.

Let R be a block in some partition. From Lemma 4.1 we know that all parameters π_R are multiples of some constant β depending on k. Thus π_R/β is an integer for each R. For each model M_1 (consisting of a partition and the maximum-likelihood probabilities assigned to its blocks), define the *pseudo-likelihood* function $\mathcal{L}(M_1)$ as follows:

$$\mathcal{L}(M_1) = \prod_R \left[\frac{\pi_R}{|R|}\right]^{\pi_R/\beta}$$

Note that $\mathcal{L}$ is a polynomial size rational number. We claim that the pseudo-likelihood function is maximized precisely on those models equivalent to M_*: i.e., $\mathcal{L}(M_*) \geq \mathcal{L}(M_1)$, with strict inequality whenever $M_* \not\equiv M_1$. From this, we conclude the proof by computing $\mathcal{L}$ for each model and choosing one with the largest such value. Failure can occur only with probability δ, should our parameter estimates be wrong. Since it is a property of every distance function that the distance **d** between equivalent models is zero, the result holds for arbitrary distance functions.

Let R be a block in the partition of M_1, $\{S_j\}$ the blocks of M_*, and for each j, $X_j \equiv R \cap S_j$. Then

$$\begin{aligned} \log \mathcal{L}(M_1) &= \beta^{-1} \sum_R \pi_R \log(\pi_R/|R|) \\ \log \mathcal{L}(M_*) &= \beta^{-1} \sum_S \pi_S \log(\pi_S/|S|) \end{aligned}$$

We take one term from $\log \mathcal{L}(M_1)$ corresponding to a particular box R and expand this as a sum over the sub-boxes $X_j = R \cap S_j$.

$$\log \mathcal{L}(M_1)_R = \beta^{-1} \sum_{X_j} \pi_{X_j} \log(\pi_R/|R|).$$

The portion of $\log \mathcal{L}(M_*)$ due to these same sets X_j is

$$\log \mathcal{L}(M_*)_R = \beta^{-1} \sum_{X_j} \pi_{X_j} \log(\pi_{X_j}/|X_j|),$$

since $\pi_{X_j}/|X_j| = \pi_{S_j}/|S_j|$ (just a scale change). Subtracting,

$$\begin{aligned}
\log \mathcal{L}(M_*)_R - \log \mathcal{L}(M_1)_R &= \beta^{-1} \sum_{X_j} \pi_{X_j} \log \left(\frac{\pi_{X_j}|R|}{\pi_R |X_j|} \right) \\
&\geq \beta^{-1} \sum_{X_j} \pi_{X_j} \left(1 - \frac{\pi_R |X_j|}{\pi_{X_j}|R|} \right) \\
&= \beta^{-1} [\pi_R - (\pi_R/R) \sum |X_j|] \\
&= 0.
\end{aligned}$$

And the inequality above is strictly > unless all $\pi_{X_j}/|X_j|$ are equal — i.e., all vectors in R have the same probability.

Summing over all the boxes R, we have $\log \mathcal{L}(M_*) \geq \log \mathcal{L}(M_1)$, with strict inequality unless the two models are equivalent. The claim follows directly by monotonicity of the log function.

□.

As a corollary, k-RDC is efficiently learnable given that we can generate and test all N partitions in polynomial (in n) time. Also, there are several ways to relax the conditions on the model class without sacrificing data efficiency. Most notably, the hard bound k can be replaced by a poly-log function of n, thus allowing the number of classes to grow (slowly) and the minimum non-zero probability π to get smaller.

But while the model family is "efficiently" learnable in theory, a generate-and-test algorithm is clearly impractical. Furthermore, by requiring that all classes be rectangular – even those containing vectors with zero probability – a large number of classes must be dedicated to boxing in vectors which never occur. This can be avoided by allowing one set in the partition to be non-rectangular, and that it be the only one with $\pi = 0$. Virtually the same argument goes through, provided some condition is imposed forcing all rectangular boxes to contain $\mathcal{O}(r^{n-\log n})$ vectors.

We have also studied less stringent classification classes, and find that typically these classes are data efficient, but that an efficient search algorithm to find the model maximizing the likelihood on the sample vectors is difficult to construct (cf. [5]).

5 Markov Models

An important feature of our model of unsupervised learning is that stochastically dependent input values can be modeled. One of the simplest ways to relax the independence assumption for inputs is to use Markov-dependent random variables. In this case, the total dependence of the t'th prediction on the previous $t-1$ inputs can be reduced to dependence on one of a finite set S of values ("states"):

$$\mathbf{P}(x_t \mid x_1 \ldots x_{t-1}) = \mathbf{P}(x_t \mid s_{t-1}), \tag{3}$$

where the state $s_{t-1} \in S$ captures all the dependency on $x_1, \ldots, x_{t-1}$.

Let A be a finite set of symbols (e.g., $\{0,1\}$). A Markov chain[3] over A consists of a (finite) set S of states, one of which is designated the current state; a deterministic transition function

[3]More precisely, a finite function of a homogeneous, discrete time, discrete state, unifilar Markov process.

δ mapping each state s and input value a into a new state s'; and probabilities $p(a \mid s), a \in A$ for these transitions such that for all $s \in S$, $\sum_{a \in A} p(a \mid s) = 1$.

A *Markov model* M is a program based upon such a Markov chain. Before the t'th input bit arrives, it uses the input x_{t-1} and current state s_{t-1} to compute the next state $s_t = \delta(s_{t-1}, x_{t-1})$, and to output the set of probabilities $\{p(a \mid s_t) \mid a \in A\}$. The parameterized distribution family Λ is a thus multinomial family (a Bernoulli family in case $|A| = 2$). But unlike models that view the environment as a sequence of independent events, M may output *different parameters* at different times. Since we are interested in modeling environments that are in equilibrium, we may restrict our models to those whose Markov chains are strongly connected – i.e., indecomposable, with no transient states [2, 8].

As a growth parameter n, we typically count the maximum number of states; but we must also limit the length in bits of the transition probabilities. A bound of length n on probabilities for an n-state machine is quite liberal since it allows exponentially small probabilities. Thus one family $\mathcal{M}_n$ of Markov models corresponds to a fixed set $A = \{0, 1\}$, the family Λ_n in Example 3.3, and the class of Markov chains with at most n states and with transition probabilities in Λ_n.

Proposition 5.1 (see [3]) *Let A be a finite set and $\mathcal{M}$ a finite class of Markov chain models over alphabet A. Then $\mathcal{M}$ is learnable in the limit.*

It is also interesting that the Markov chain model with the minimum number of states can be learned in the limit ([13, 15]).

Besides illustrating a dependent-variable model class, Markov families provide us with an example of a family that is not efficiently learnable. Finite automata are notoriously difficult to learn in the supervised case [1, 9, 14], so it is not surprising to discover that they are also hard to learn in the unsupervised case. We give a simple proof for the $\mathbf{d}_1$ distance function. ($\mathbf{d}_2$ is a bit more complicated; see [13]).

Theorem 5.2 The family $\mathcal{M}_n$ of Markov models (as defined above) with at most n states and transition probabilities bounded in length by n bits is not data efficient (and hence not efficiently learnable) with respect to $\mathbf{d}_1$.

PROOF: We exhibit two models which a polynomial-size sample cannot distinguish with arbitrarily high probability, even though the two models each have more than ϵ error relative to one another.

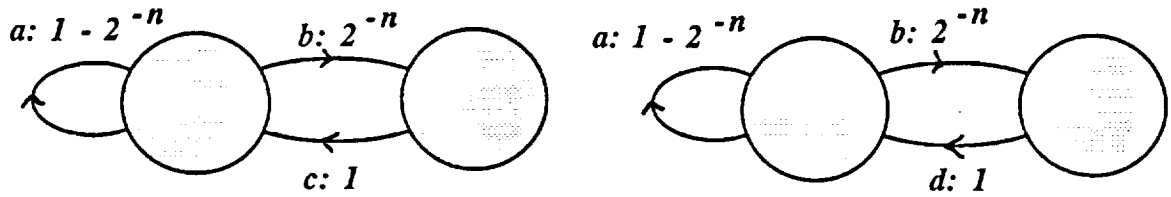
Assume an alphabet $A = \{a, b, c, d\}$. Machines M_1 and M_2 are shown below. Until a b transition occurs, the predictions for the two models M_1 and M_2 are identical. The probability that no b-transition occurs in t transitions is $(1 - 2^{-n})^t$, and we require t such that this probability is less than δ. Taking logs, we find:

$$t \cdot \log(1 - 2^{-n}) < \log \delta.$$

But if t is bounded by a polynomial in n, this condition fails for all sufficiently large n.

If no b transition occurs, any algorithm must choose between models M_1 and M_2 on the basis of a string of a's alone. If it chooses M_1 but the actual target is M_2, the error is 1 (the

difference in probability for the c transition out of the rightmost state). If it chooses M_2 but the actual target is M_1, the error is again 1 based on the d transition.



The algorithm could still succeed if it proposed some further model M_3 different from either M_1 or M_2, such that $\mathrm{d}(M_3 \mid M_1) < \epsilon$ and $\mathrm{d}(M_3 \mid M_2) < \epsilon$. But we can show that there is no such model. Immediately after the first b transition, suppose M_3 is in a state that predicts a c with probability $p_c < 1$ and a d with probability $p_d < 1$. Necessarily $p_c + p_d \leq 1$, and so if $p_c \geq \frac{1}{2}$ then $p_d \leq \frac{1}{2}$. Thus either $\mathrm{d}(M_3 \mid M_1)$ or $\mathrm{d}(M_3 \mid M_2)$ is at least $\frac{1}{2}$. Hence if $\epsilon < \frac{1}{2}$, no model is within ϵ of both M_1 and M_2. □

6 Concluding Observations

We have cast the problem of unsupervised learning into a statistical format that differs from that of conventional pattern recognition in its emphasis on efficient prediction and its concern for timely convergence ("learning"). Starting from definitions of models and families of model classes, we considered successively stronger concepts of learnability:

1. *learnability in the limit*: disregarding computational cost, a correct model can be identified in finite time.

2. *data-efficient learnability*: uniformly learnable from a feasibly small amount of data, without consideration of the computational resources required to extract the information from that data.

3. *efficient learnability*: uniformly learnable with a feasibly small data sample that can be processed using polynomially small computing resources.

To illustrate these concepts, we have developed and analyzed three illustrative model families: Bernoulli families, k-RDC classification families, and a particular Markov family. Much work remains to be done in constructing meaningful, expressive, and useful model families, algorithms to learn them, and techniques for analyzing their complexity.

From experience practical learning algorithms need to be "on-line" in the sense that they successively refine their hypotheses as more data arrives, instead of collecting a large clump of data and emitting a single hypothetical model. If the storage resources of the model are bounded, then the predictions of the model will exhibit a random-walk behavior as runs of typical and atypical input data are received. Related problems concern the performance of such an algorithm when the source of the data is not a model in the target class, and when the best model may change ("drift") with time.

While this work was intended to extend the Valiant model of supervised learnability, it is worth a careful look at the differences. To that end, let us define a *supervised-modeling problem*

as follows. Suppose $\mathcal{F}$ is a set of *classification functions* defined on a finite vector space X. The purpose of these functions is to associate each vector $\mathbf{x}$ with one of a finite set of pre-determined classes $C_1, \ldots, C_k$. However, a function $f \in \mathcal{F}$, rather than assigning a unique class to a vector, specifies a *probability distribution*: $f(\mathbf{x})$ is a set of k rational values $p_1(\mathbf{x}), \ldots, p_k(\mathbf{x})$ summing to one, with the intended interpretation that $\mathbf{x}$ belongs to class C_i with probability $p_i(\mathbf{x})$. The learning problem is to identify an arbitrary function $f_* \in \mathcal{F}$, given a stream of input examples that have been classified by a teacher. The teacher provides examples of $f_* \in \mathcal{F}$ by choosing a vector $\mathbf{x}$ (not necessarily independently) from X according to some unknown sampling scheme, assigning a classification C_i to x according to the probability distribution $f_*(\mathbf{x})$, and presenting the pair $(\mathbf{x}, C_i)$ to the learning algorithm.

In the special case where the vector space is $\{0,1\}^n$, and there are exactly two classes, and all the probabilities are either zero or one, and the sampling scheme consists of independent trials, we then have the problem of learning Boolean concepts. The Valiant model is obtained by introducing a distance function $\mathrm{d}(f_1 \mid f_2)$ and requiring that the distance between the target and the hypothesized functions be less than ϵ with high probability $(1 - \delta)$. The distance (or "error") between f and f_* is the measure of the set $f \triangle f_*$ of vectors on which f and f_* yield different classifications, using the probability distribution over X. Let E denote the mathematical expectation of a function with respect to the distribution $\mathbf{P}$ on X. Then

$$\begin{aligned} \mathrm{d}(f \mid f_*) &= \mathsf{E}[f(\mathbf{x}) - f_*(\mathbf{x})]^2 \\ &= \mathrm{d}_3(f \mid f_*), \end{aligned}$$

the mean-squared error defined above (Example 3.10). We thus see that the Valiant model is a highly specialized case of model learning over a vector space $X \times \{0,1\}$, in which the models need not account for the distribution of the X part of the input data, but only for their classification, with the mean-squared error d_3 as the measure of distance between hypotheses. From this perspective, many interesting generalizations of the model – including that of non-independent examples – become possible.

7 Acknowledgments

Discussions with Dana Angluin, Peter Cheeseman, Evan Gamble, David Haussler, Don Kimber, Nick Littlestone, Alex Milosavljevic, Matthew Self, John Stutz, and Manfred Warmuth have been very helpful in developing these ideas. Hamid Berenji, Peter Friedland, Bill Gevarter, Phil Nachtsheim, and my colleagues at NASA Ames Research Center have also provided helpful comments.

References

[1] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

[2] R. B. Ash. *Information Theory*. J. Wiley, New York, 1965.

[3] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat.*, 41:164–171, 1970.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[6] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* J. Wiley, 1973.

[7] B. S. Everitt and D. J. Hand. *Finite Mixture Distributions.* Chapman and Hall, 1981.

[8] W. Feller. *An Introduction to Probability Theory and its Applications, 3rd Edition.* J. Wiley, New York, 1968.

[9] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc.*, 58:13–30, 1963.

[11] B.-H. Juang and L. Rabiner. A probabilistic distance measure for hidden markov models. *AT&T Technical Journal*, 64(2):391–407, 1985.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:79–86, 1951.

[13] P. Laird. *Learning by Making Models.* Technical Report RIA-88-4-12-0, NASA-Ames Research Center, AI Research Branch, 1988.

[14] L. Pitt and M. Warmuth. Reductions among prediction problems: on the difficulty of predicting automata. In *3rd Structure in Complexity Theory Proc.*, 1988.

[15] S. Rudich. Inferring the structure of a markov chain from its output. In *Proc. 26th FOCS*, pages 321–325, 1985.

[16] J. E. Shore and R. W. Johnson. Properties of cross-entropy minimization. *IEEE Trans. Inf. Th.*, IT-27:472–482, 1981.

Ames Research Center

REVIEW AND AUTHORIZATION RECORD

NASA

INSTRUCTIONS

ARC 310 is used for Ames scientific and technical publications (as defined in "Guidelines for Scientific and Technical Publications") authored or coauthored by Ames employees – namely, Technical Papers (TP), Reference Publications (RP), Special Publications (SP), Journal Articles (JA), Meeting Papers (MP), some Technical Memoranda (TM), and some Conference Publications (CP). The author must specify, on the back of this form, manuscript editing and/or typing required of the Publications Office. Use ARC 311 for Ames scientific and technical publications for rapid printing with limited distribution.

MANUSCRIPT IDENTIFICATION

RTOP | Job Order | Org Code | ATP File

Efficient Unsupervised Learning
Title

LAIRD, PHILIP D.
Author(s)

Publication Category: ☐ TP ☐ RP ☐ SP ☐ JA ☐ MP ☐ TM ☐ CP

☒ None ☐ Confidential ☐ Secret
Security Classification

COMPUTATIONAL LEARNING THEORY PROCEEDINGS
If JA or MP, specify name of journal or meeting.

REVIEWERS' APPROVAL OF DRAFT

Manuscript satisfies Criteria for Publication Numbers 1 and 2 on reverse side.

Evan A. Gamble — Signature — 6/23/88 Date

Matthew Sell — Signature — 6/23/88 Date

BRANCH CHIEF APPROVAL OF DRAFT

Manuscript satisfies Criteria for Publication Numbers 1-5 on reverse side.

Peter Friedland — Signature — 6/26/88 Date

DIVISION CHIEF APPROVAL OF DRAFT

Manuscript satisfies Criteria for Publication Numbers 1-5 on reverse side.

Signature — Date

AUTHOR APPROVAL OF EDITING

Manuscript has been edited according to my instructions on reverse side and is ready for final typing.

Signature — Date

AUTHOR, DIVISION CHIEF, ORGANIZATIONAL DIRECTOR APPROVAL OF PROOF COPY

Manuscript is correct and may be printed and distributed. (For a journal article or meeting paper, only the author need approve.)

Philip Laird — Signature — Author — 6/24/88 Date

Signature — Division Chief — Date

Signature — Organizational Director — Date

PUBLICATIONS OFFICE CERTIFICATION

Manuscript conforms to NASA publication policy and is suitable for microfilming in accordance with Government standards.

Signature — Chief, Publications Office — Date

ARC 310 (Rev. Feb 80)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
Dates attached

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE

Titles/Authors - Attached

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Code FIA - Artificial Intelligence Research Branch
Information Sciences Division

8. PERFORMING ORGANIZATION REPORT NUMBER

Attached

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Nasa/Ames Research Center

Moffett Field, CA. 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Available for Public Distribution

Peter Friedland 5/14/92 BRANCH CHIEF

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstracts ATTACHED

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT