Quantifying Prior Determination Knowledge using the PAC Learning Model

Sridhar Mahadevan
IBM T.J. Watson Research Center
Box 704, Yorktown Heights
NY 10598
(sridhar@watson.ibm.com)

Prasad Tadepalli
Department of Computer Science
Oregon State University
Corvallis, OR 97331
(tadepalli@cs.orst.edu)

Abstract

Prior knowledge, or bias, regarding a concept can speed up the task of learning it. Probably Approximately Correct (PAC) learning is a mathematical model of concept learning that can be used to quantify the speed up due to different forms of bias on learning. Thus far, PAC learning has mostly been used to analyze *syntactic* bias, such as limiting concepts to conjunctions of boolean prepositions. This paper demonstrates that PAC learning can also be used to analyze *semantic* bias, such as a domain theory about the concept being learned. The key idea is to view the hypothesis space in PAC learning as that consistent with *all* prior knowledge, syntactic and semantic. In particular, the paper presents a PAC analysis of *determinations*, a type of relevance knowledge. The results of the analysis reveal crisp distinctions and relations among different determinations, and illustrate the usefulness of an analysis based on the PAC model.

Keywords: Determinations, PAC Learning, Bias, Prior Knowledge, Incomplete Theories.

Running Head: Quantitative Analysis of Determination Knowledge.

1 Introduction

Prior knowledge or bias [Mitchell, 1980] regarding a concept can sometimes dramatically reduce the number of examples needed to learn it. One common form of bias is *syntactic* constraints on the concept description language. For example, if a learner knows that the concept being learned is describable by a purely conjunctive boolean expression, a special technique for inducing such expressions can be used to expedite the learning. There have been many successful attempts at quantifying syntactic bias, such as [Haussler, 1988]. These approaches are based on a mathematical model of concept learning called Probably Approximately Correct (PAC) learning introduced by Valiant [Valiant, 1984]. For a detailed introduction to the model, see [Natarajan, 1991].

However, humans, and indeed some machine learning systems, draw their power not only from syntactic bias, but also from knowing something about the content of the particular concept being learned. For example, a human learning to play a new game often uses his general knowledge of competitive games to accelerate learning. Similarly, machine learning programs like FOCL rely on a "domain theory" to expedite learning new knowledge [Pazzani, 1992]. In other words, these systems exploit a "semantic bias" in addition to a syntactic bias.

Quantifying semantic bias or prior knowledge is an important problem in artificial intelligence (AI). In a recent introductory book on AI [Rich and Knight, 1991], after discussing Valiant's model of PAC learning, the authors of the book note (pages 482-483):

> After all, people are able to solve many exponentially hard problems by using knowledge to constrain the space of possible solutions. Perhaps mathematical theory will one day be used to quantify the use of such knowledge, but this prospect seems far off.

In this paper we show that the authors' pessimism is somewhat unwarranted, and that semantic bias can be quantified using essentially the same PAC learning framework used to analyze syntactic bias. To our knowledge, the work described here – which was first reported in [Mahadevan and Tadepalli, 1988] – represents one of the first attempts to analyze semantic bias using PAC learning. Russell's work on "tree-structured bias" is another early example of such an analysis [Russell, 1988].

PAC learning is based on a paradigm wherein a teacher provides a learner with examples of a target function from an initially agreed upon space of possible functions. This space can be viewed as representing the syntactic bias of the learner. Examples are selected randomly according to a fixed but arbitrary probability distribution unknown to the learner. The task of the learner is to find with high probability a function that is a good approximation of the target function – hence the name "Probably Approximately Correct" learning. The learner prunes the function space by eliminating functions that are inconsistent with the examples. Learning is complete when the only functions that remain unpruned are with high probability good approximations of the target function. The more restricted the initial space of functions that contains the target function, the fewer the functions that have to be pruned to learn it, and hence, the fewer the examples needed to do the pruning. In general, the number of examples needed to learn an arbitrary function in a function space increases monotonically with the number of functions in the function space.

In order to obtain broadly applicable results, any attempt to quantify semantic bias should be insensitive to particular ways of representing the prior knowledge. The key observation behind this paper is that such an analysis can be achieved by modeling knowledge abstractly as a space of functions of which the target function is a member. We thereby generalize the notion of function space in PAC learning to the set of functions consistent with *all* prior knowledge – both syntactic and semantic.

Our results are based on the central results in PAC learning that imply that the number of examples needed for robust learning increase with some measure of the complexity of the function space. Since a reliable learning algorithm has to learn any and all functions consistent with its prior knowledge, our negative results, which are based on the size of the function space, are hard lower bounds.[1] They imply that certain kinds of prior knowledge are not strong enough to make a learner converge after seeing a reasonably small number of examples, whatever form that knowledge is represented in. In contrast, our positive results are constructive in that they are accompanied by polynomial-time learning algorithms.

An analysis of the effect of a particular piece of domain knowledge on learning may not be useful in other domains. Hence, we analyze the usefulness of general *forms* of knowledge in a domain-independent way. In particular, this paper presents an analysis of *determinations*, a general form of relevance knowledge. Relevance knowledge consists of information about the dependence among different features. A feature P is relevant to another feature Q if the fact that P holds for some object affects whether Q also holds for that object. Determinations were originally proposed by Davies and Russell [Davies and Russell, 1987, Russell, 1986, Russell, 1989] in the context of analogical reasoning. An example of a determination is the prior knowledge that "nationality" determines "language", that is, individuals of the same nationality speak the same language. This particular form of determination can be weakened in several ways. For example, another form of determination allows for individuals with the same nationality to speak different languages, as long as they share a common language. Yet another form allows for a small number of "exceptional" individuals who may not speak any common language, and so on.

We analyze each of these forms of determinations. In particular, for each type of determination, we study its effect on learning a function by comparing the number of examples required in the absence and presence of the determination. Several interesting facts emerge from the analysis. Minor changes in the definition of a determination can result in dramatically different learnability properties. This allows the various determinations to be ranked according to their effect on the learning process. Furthermore, apparently dissimilar determinations are actually quite similar in terms of their effect on learning.

We believe our theoretical results have direct relevance to implementors of practical knowledge-based learning systems. For example, Explanation-Based Learning (EBL) is a knowledge-intensive learning technique that relies on its ability to classify an instance using a theory of the domain [Mitchell *et al.*, 1986, Dejong and Mooney, 1986]. One of the open

[1] Negative results using the PAC learning model should be interpreted as "worst-case" theorems similar to the NP-completeness results in computational complexity theory.

problems in EBL arises when the domain theory is not adequate to classify every instance [Mitchell *et al.*, 1986]. Most approaches to this "incomplete theory problem" are based on using pre-classified training examples to expose and fill in missing parts of the domain theory [Hirsh, 1989, Hall, 1988, Mahadevan, 1989, Danyluk, 1989]. For example, one approach involves using determinations to represent gaps in the domain theory, which are filled by extracting implicative rules from the determinations [Russell, 1987, Mahadevan, 1989]. A PAC analysis can be used to determine whether the gaps in a domain theory are "small" enough so that they can be filled with a reasonably small number of examples.

The rest of this paper is organized as follows. Section 2 informally explains our approach to quantifying semantic bias. Section 3 describes the PAC learning framework. The main results on learnability of function spaces in the presence of the various determinations are given in Section 4. Section 5 discusses some implications of our formal results. Section 6 summarizes the main results of the paper.

2 Informal Overview of the Approach

In this section we informally characterize our approach to quantifying semantic bias. Suppose an intelligent agent is faced with the task of learning from examples some unknown function, such as a mapping from individuals to languages. Each example describes an individual using a set of attributes such as his or her height, weight, nationality, place of employment etc., and also lists his or her language. Any information the learner has about the unknown function before seeing the examples is its "prior knowledge." Although prior knowledge can also take the form of a "simplicity" preference ordering on the functions in the function space, in this paper we restrict ourselves to prior knowledge which constrains the set of allowed functions to a subset of all possible functions. In the absence of any such prior knowledge about the target function, the agent can do no better than storing each example. This rote learning strategy becomes prohibitive if the number of individuals (more precisely, the number of possible descriptions of individuals) is very large.

On the other hand, suppose the agent has prior knowledge in the form of a determination that any two people of the same nationality speak the same language. Given this piece of knowledge, the initial learning problem is now reduced to one of learning a function that maps nationalities to languages. If there are very few nationalities compared to the number of people, which happens to be true of our world, the learning problem is now a much simpler one. In particular, the agent can justifiably generalize from a single example: once it knows the language spoken by some individual of a given nationality, it can form a general rule stating that every individual of that nationality speaks this language.

Pursuing the nationality example further, we note that even with the prior knowledge, the learning problem is not trivial since there may be many functions that are consistent with the knowledge. For example, the function that assigns all Americans the English language, the function that assigns all Americans the Spanish language etc., are all consistent with the prior knowledge. Generally, prior knowledge will define a *space* of functions, and examples help refine the space of functions to the one target function that the learner is supposed to acquire.

Generalizing from the above example, Figure 1 illustrates the relation between the amount of prior knowledge available and the size of the function space. Given no knowledge, the space of possible functions is large, and learning requires too many examples. When some knowledge is available about the function being learned, the space is reduced since all functions that are inconsistent with the given knowledge are eliminated.

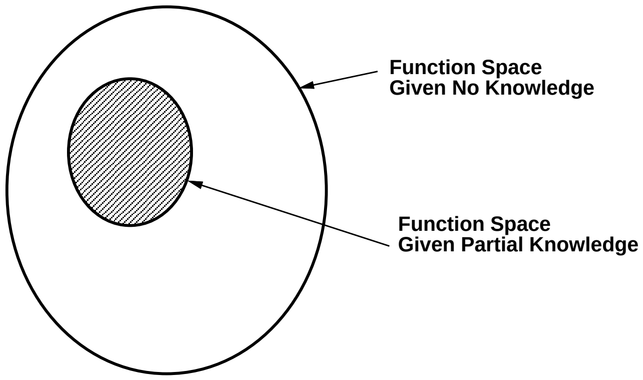

Figure 1: Reducing a function space using prior knowledge

Leaving the formal details to future sections, it will be useful here to briefly outline the form of our analysis. Given a particular determination, we estimate the size of the function space consistent with the determination. The main result in PAC learning that we use is the *dimensionality theorem*, which relates the number of training examples needed for successful learning to the size of the function space [Blumer *et al.*, 1989, Natarajan, 1989]. Informally, this theorem says that the number of examples sufficient for successful learning varies logarithmically with the asymptotic size of the function space. We apply the dimensionality theorem to the reduced function space consistent with the prior knowledge and determine bounds on the number of examples needed to learn functions in that space. If this bound is too high, that is, exponential in the problem parameters, then it is not feasible to learn that space. If this bound is reasonable, that is, polynomial in the problem parameters, then we conclude that it is feasible to learn this function space.

A practical learning technique not only needs to converge with a reasonable number of examples, but also needs to be computationally efficient. While prior knowledge will always reduce the number of examples sufficient for learning, it might sometimes increase the time complexity of searching for a function consistent with it [Haussler, 1988]. In those cases, it may be appropriate to ignore some prior knowledge and consider a bigger function space than is necessary, thus requiring a few more examples while gaining computational tractability. For each of our function spaces that can be learned with a reasonable number of examples, we isolate conditions under which they are learnable in reasonable time, and describe efficient (polynomial-time) learning algorithms for them.

3 The PAC Learning Model

In this section we give a brief overview of the relevant formal results from PAC learning. In particular, we will use a generalization of Valiant's original model to function learning studied by Natarajan [Natarajan, 1989, Natarajan, 1991].

3.1 Preliminaries

Since any domain/range element of a function can be encoded as a binary string, without loss of generality we consider learning functions from binary strings to binary strings. An *example* of a function f is a pair $(x, f(x))$. We assume a routine EXAMPLE, which outputs an example of a function f according to some fixed, but unknown, probability distribution P. In other words, the probability of a particular example $(x, f(x))$ being generated by a call of EXAMPLE is $P(x)$.

In the following, we denote the length, or the number of significant bits of string x by $|x|$. We let $\sum^n$ refer to the set of strings of length n and $\sum^*$ to the set of strings of arbitrary length. We let $Trim(w, n)$ denote the n-length prefix of string $w \in \sum^*$.

Definition 1 *A* **space of functions** F *is a set of functions from* $\sum^*$ *to* $\sum^*$*.*

The following definition limits the functions being considered to those whose output is at most a polynomial in the size of their input.

Definition 2 *If* $k(n)$ *is a fixed polynomial function, called the scale-up function, the* n^{th}-**subspace** F_n *of* $F = \{f_1, \ldots, f_i, \ldots\}$ *is* $\{g_1, \ldots, g_i, \ldots\}$ *where each* $g_i : \sum^n \to \sum^{k(n)}$ *is such that* $g_i(Trim(w, n)) = Trim(f_i(w), k(n))$ *if all* $w \in \sum^*$ *with the same* n*-length prefix are mapped by* f_i *to strings with the same* $k(n)$*-length prefix, and undefined otherwise.*

A simple example will help illustrate these definitions. Assume that the task is to learn boolean functions. The function space B is the set of all possible boolean functions which output a single bit. The n^{th}-subspace B_n is a restriction of B to functions over input bit strings of length n. Note that B_n has 2^{2^n} functions.

3.2 PAC Learning

We now formally describe the PAC learning model. For convenience, we distinguish learning that converges with reasonable (polynomial) number of examples, which we call *feasible learnability*, from learning that also bounds the computational time, which we call *polynomial-time learnability.*

Definition 3 *A space of functions* F *is* **feasibly learnable** *if there exists an algorithm* A *that, given an error parameter* ϵ*, a confidence parameter* δ*, and the problem size* n*,*
(i) makes calls to EXAMPLE, whose number is polynomial in n*,* $\frac{1}{\epsilon}$*, and* $\frac{1}{\delta}$*, and*

(ii) for all functions f in F_n and all probability distributions Pr, with probability at least $1-\delta$ outputs a function g such that,

$$\sum_{x \in S} Pr(x) \leq \epsilon$$

where $S = \{x \mid x \in \sum^n$ and $f(x) \neq g(x)\}$.
We make no assumptions on the representation of g other than that there exists a polynomial time algorithm that, given g and x, outputs $g(x)$.

Under the above conditions, A is called a **learning algorithm** *for F.*

The parameter ϵ specifies the error of the function g when compared to the real function f the learner is trying to approximate. The error is measured by the probability that f and g differ on some example chosen randomly using the same distribution P that was used during the learning. Since the approximation is obtained using randomly chosen training examples, they might sometimes be unrepresentative, in which case the approximation learned from them may not be sufficiently accurate on representative test examples. A learning algorithm must ensure that the probability of this event is lower than the confidence parameter δ.

Note that we do not require the output function g to be in F_n. In other words, we allow the learner to output a function which violates the prior knowledge, as long as it approximately agrees with the target function with a high probability on the training distribution. The advantage of this definition is that it avoids the problem of having to check that g is consistent with the prior knowledge, which could sometimes be computationally complex [Haussler *et al.*, 1988, Pitt and Valiant, 1988]. This definition of learnability is also called "predictability" in PAC learning literature [Haussler *et al.*, 1988, Natarajan, 1991].

To study the time requirements of learning, we need to assume a representation or index for the functions. Since each function may have multiple names, the index maps functions to sets of binary strings.

Definition 4 *An* **index** *of the function space F is a function $I : F \to 2^{\Sigma^*}$, such that $\forall f, g \in F$, if $f \neq g, I(f) \bigcap I(g) = \{\}$.*

Definition 5 *A function class is said to be* **polynomial-time learnable** *if there is a learning algorithm that runs in time polynomial in n, the length of the shortest index of the target function $f \in F_n$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.*

3.3 Identification and Dimensionality

The following additional definitions are needed to state the main theorems from learnability theory.

Definition 6 *A function f is* **consistent** *with a set of examples S if $(x, y) \in S \Rightarrow f(x) = y$.*

Typically, learning algorithms work by guessing a function which is consistent with all the input examples. Following [Rivest, 1987] we call such an algorithm an *identification.* Formally,

Definition 7 *An* **identification** *O of a space of functions F is an algorithm that takes as input an integer n and a set of examples $S = \{(x_i, y_i)\}$, where each x_i is of length at most n, and produces an output function $f \in F$ that is consistent with S, if such exists. If O runs in time polynomial in the length of its input and the length of the shortest index of the functions consistent with S, we say that F is* **polynomial-time identifiable**.

An identification for the boolean function space B will take as input many examples of the form (x_i, y_i), where x_i is a bit string of length n and y_i is 0 or 1, and outputs the following function f. f outputs a 1 for any input string x_i in the example set such that $y_i = 1$, and outputs a 0 on all other inputs. f can be represented simply by the set of positive instances, that is, examples for which the output is a 1. Since f can be produced in time polynomial in the number of examples, B is polynomial-time identifiable.

We now introduce Natarajan's notion of "dimension," a measure of the size of a function space [Natarajan, 1989]. The relationship of Natarajan's dimension to the more popular Vapnik-Chervonenkis dimension [Blumer *et al.*, 1989] is discussed in [Natarajan, 1989].

Definition 8 *The* **dimension** *of F_n, the n^{th} subspace of F, is* $\log_2 |F_n|$.

Definition 9 *A space of functions F is of dimension $D(n)$ if, for all n, the dimension of F_n (the n^{th} subspace of F) is $D(n)$. If there is a polynomial $p(n)$ such that $D(n) \leq p(n)$ for all n, F is said to be of* **polynomial dimension**.

To calculate the dimension of the function space B in our boolean function example above, we note that there are 2^{2^n} possible functions in B_n. Thus, the dimension of the function space B is $D(n) = 2^n$. Now consider a subspace B' of B in which every boolean function maps exactly one input string to 1, and the rest to 0. It is easy to see that there are only 2^n functions in B'_n, one function for each input string. The dimension of this new function space B' is the polynomial $D(n) = n$.

3.4 Learnability Theorems

The main results we will be using from the theory of PAC learnability can now be stated [Natarajan, 1989].

Theorem 1 (Natarajan) *A space of functions F is feasibly learnable if and only if it is of polynomial dimension.*

Theorem 2 (Natarajan) *A space of functions is polynomial-time learnable if it is of polynomial dimension and is polynomial-time identifiable.*

Taking our boolean function example once again, since the dimension of the function space B is exponential, it is not feasibly learnable. But if the learner has the additional knowledge that the target function maps exactly one input string to 1 and the rest to 0, we can simply focus on the reduced function space B', which has a polynomial dimension, and feasibly learn it. B' is also polynomial-time learnable because the same identification that we discussed before for B would work for B' as well, and runs in time polynomial in the length of its input. This example clearly illustrates how a single piece of knowledge, such as the existence of a single positive instance for the target function, can make a dramatic difference to the learnability of the function space.

The following theorem allows us to estimate the exact number of examples sufficient to learn a function given the dimensionality of the space containing it.

Theorem 3 (Natarajan) *If $Dim_F(n)$ is the dimension of a function space F, then any algorithm which collects and identifies a set of examples of size $\frac{1}{\epsilon}(Dim_F(n)\ln 2 + \ln_e(\frac{1}{\delta}))$ is a learning algorithm of F.*

The proof for the above theorem follows a similar result for concept learning given in [Blumer *et al.*, 1989] or [Natarajan, 1987].

4 Learnability Results

This section describes the main results of this paper on quantifying relevance knowledge defined by various determinations.

4.1 Determinations

Determinations are intended as a formalization of the notion of *relevance*. Intuitively, an attribute P is relevant to an attribute Q if the fact that P holds for some object affects whether Q holds of that object. For example, the fact that the attribute American-Nationality holds for a certain individual affects whether the attribute Speaks-English holds true for him or her. On the other hand, we feel reasonably certain that the Height attribute will not similarly affect the Speaks-English attribute.

The simplest type of determinations are called *total* determinations. Russell introduced five types of total determination in his thesis [Russell, 1986]. The first of these is defined as follows:

Definition 10 *Let $P(x, y)$ and $Q(x, z)$ be any two first-order sentences, where x represents the set of variables that occur free in both P and Q, while y and z represent the set of free variables that occur only in P and Q, respectively. We say $P(x, y) \succ Q(x, z)$ iff*

$$\forall w, x[[\exists y P(w, y) \land P(x, y)] \Rightarrow \forall z[Q(w, z) \Leftrightarrow Q(x, z)]]$$

An example (which we will use as a running example throughout this paper) will help clarify the above definition. Let $P(x, y)$ denote the predicate $Nationality(x, y)$, which means that the individual x has nationality y. Also let $Q(x, z)$ denote the predicate $Language(x, z)$, which means that x speaks language z. Then, the above total determination states that if there exist two individuals x and w who share a nationality y, then x and w will speak the same set of languages.

Determinations can be viewed as a form of incomplete knowledge [Russell, 1987]. For example, from

$$Nationality(x, y) \succ Language(x, z)$$

and

$$Nationality(John, US) \wedge Language(John, English)$$

it follows that

$$\forall x Nationality(x, US) \Rightarrow Language(x, English)$$

However, just knowing that nationality determines language is not sufficient to compute an individual's language from his nationality. Examples are required to fill in this knowledge, and thus they are a source of new information (unlike the situation in EBL where examples are a logical consequence of the domain theory [Mitchell *et al.*, 1986]). In general, from $P(x, y) \succ Q(x, z)$ and $P(A, B) \wedge Q(A, C)$, the implication $\forall x P(x, B) \rightarrow Q(x, C)$ follows.

4.2 Function Space Consistent with a Determination

Let $P(x, y)$ and $Q(x, z)$ be any two first-order formulas specified as part of a determination $P(x, y) \succ Q(x, z)$, where, as before, x represents the set of free variables appearing in both P and Q, and y and z represent the set of free variables appearing only in P and Q, respectively. For some sets I, N, and L, let $\mathcal{P} \subseteq I \times N$ and $\mathcal{Q} \subseteq N \times L$ denote the extensions of the predicates P and Q respectively. Therefore, the variables x range over I, the variables y range over N, and the variables z range over L. In terms of the nationality example, I is the set of individuals, N is the set of nationalities, and L is the set of languages.

We denote the set $\{y \mid P(x, y)\}$ by P_x, and the set $\{y \mid Q(x, y)\}$ by Q_x. The task is to learn to predict Q_x, given x and P_x. We view this as learning a function from $D = \{\langle x, P_x\rangle : x \in I\}$ to 2^L. Let F denote the set of all such functions $f_{P,Q} : D \rightarrow 2^L$ for a given $\mathcal{P}$. The training examples consist of the input-output pairs $(\langle x, P_x\rangle, Q_x)$.

Any particular relations $\mathcal{P}$ and $\mathcal{Q}$ uniquely define a function $f_{P,Q} \in F$ such that, for all $x \in I$, $f_{P,Q}(\langle x, P_x\rangle) = Q_x$. A determination $P(x, y) \succ Q(x, z)$ can be viewed as a constraint on the relations $\mathcal{P}$ and $\mathcal{Q}$. With every determination $P(x, y) \succ Q(x, z)$, we can associate a space of functions $F_{\succ} = \{f_{P,Q}\} \subseteq F$, defined by all particular relations $\mathcal{P}$ and $\mathcal{Q}$ satisfying that determination. We call it the space of functions *consistent with* or *defined by* that determination. Formally, $F_{\succ} = \{f_{P,Q} : P(x, y) \succ Q(x, z)\}$.

An example will help clarify the above definitions. Consider the determination $Nationality(x, y) \succ Language(x, z)$. Let I = {Giuseppe, John, Lisa, Isabella, Mami }, N = {Italy, US, Japan }, and L = {Italian, English, Japanese }. Further, let $\mathcal{P}$ = {(Giuseppe, Italy), (John, US), (Lisa, US), (Isabella, Italy), (Mami, Japan)}. For the above $\mathcal{P}$, it is easy to see that $\mathcal{Q}$ = {(Giuseppe, Italian), (John, English), (Lisa, English), (Isabella, Italian), (Mami, Japanese)} is consistent with the above determination, whereas $\mathcal{Q}'$ = {(Giuseppe, Japanese), (John, English), (Lisa, English), (Isabella, Italian), (Mami, Japanese)} is not. The reason, of course, is that, Giuseppe and Isabella, who are both from Italy, are mapped to the same language by $\mathcal{Q}$, but mapped to two different languages by $\mathcal{Q}'$. Hence the function $f_{P,Q}$ with mappings $\langle$ Giuseppe, {Italy} $\rangle \rightarrow$ {Italian}, $\langle$ John, {US}$\rangle \rightarrow$ {English}, etc. is in $F_{\succ}$, and the other function $f_{P,Q'}$ with mappings $\langle$ Giuseppe, {Italy} $\rangle \rightarrow$ {Japanese}, and $\langle$ Isabella, {Italy} $\rangle \rightarrow$ {Italian}, etc. is not.

As we said before, the nationality determination makes the function learning problem feasible because it reduces the original problem of learning a function that maps individuals to languages, which is infeasible, to a simpler problem, namely learning a function from nationalities to languages. We can view the domain of the new function, that is, nationalities, as an *abstraction* of the domain of the old function, that is, individuals. Figure 2 illustrates this point, showing how individuals sharing a nationality can be abstracted by their nationality. In Figure 2, John and Lisa are grouped together as Americans, and Giuseppe and Isabella are grouped as Italians. Now, learning a mapping from nationalities to languages effectively permits us to predict a person's language by knowing his/her nationality. The amount of abstraction achieved by a determination depends on the number of nationalities and individuals, and will turn out to be the basis for our learnability results. For the above determination, learning becomes feasible when the number of nationalities is much smaller than the number of individuals (we make this statement more precise below).

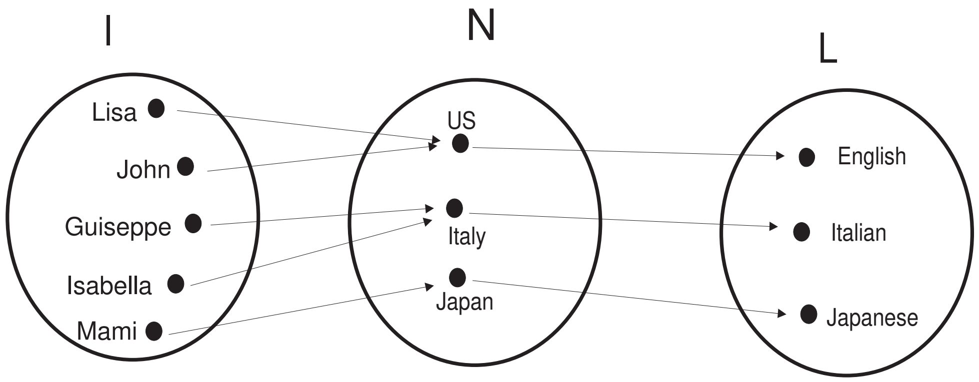

Figure 2: Abstracting the domain of a function using a determination

In order to quantify the amount of abstraction achieved by a determination, we have to parameterize the functions. We do this by parameterizing the sizes of various components of the training examples, indirectly bounding the sizes of the various sets and functions involved. In any training example $(\langle x, P_x\rangle, Q_x)$, we assume that $|x| = n$, $|P_x| = c$, and $|Q_x| = l$. Since x is any member of I, these assumptions imply $|I| \leq 2^n$. On the other hand, since P_x is any subset of N and Q_x is any subset of L, it follows that $|N| \leq c$ and $|L| \leq l$.

We make the assumption that l varies as a polynomial function of n, and is the scale-up function of F. So the n^{th}-subspace F_n of F can be defined as $\{g : \sum^n \rightarrow \sum^{l(n)} \mid \forall x \in I, g(x) = f(\langle x, P_x\rangle)$, where $f \in F\}$.

We can now get bounds on the size of the function subspace F_n as a function of n and l. Note that each function in F_n is from I, a set of size $\leq 2^n$, to 2^L, a set of size $\leq 2^l$. Hence, in the worst case, we have $|F_n| = (2^l)^{2^n}$, and $dim(F) = \log |F_n| = 2^n l$. Using the dimensionality theorem, F is not learnable without some additional prior knowledge, since it is of exponential dimension in n.

4.3 Results on Total Determinations

The first set of results concern the various types of total determinations. For each type of determination we compute bounds on the dimensionality of the function space consistent with that determination. Using the learnability theorems presented above, we can then determine the learnability of each space. We present detailed proofs for two cases in this section and refer the reader to the appendix for the rest.

A function in the function space defined by the determination $P(x,y) \succ Q(x,z)$ is illustrated in Figure 3. As a generalization of the example discussed in previous section, individuals may have multiple P values, that is, nationalities. Each ellipse in the domain of the function in Figure 3 represents a nationality. Individuals who share at least one nationality speak exactly the same set of languages.

The following theorem affirms the polynomial-time learnability of function spaces consistent with the total determination $\succ$.

Theorem 4 *The space of functions $F_{\succ}$ consistent with a determination $P(x,y) \succ Q(x,z)$ is polynomial-time learnable if $|range(P)| \le c$ and $|range(Q)| \le l$ are polynomials in $|x| = n$.*

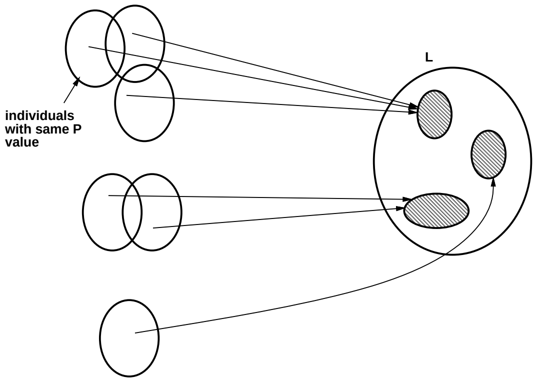

Figure 3: Part of a function in the space consistent with the $\succ$ determination

Proof: Let us define a relation R such that any two elements a and b in I are related by R iff the sets P_a and P_b are not mutually exclusive. It follows from the definition of the $\succ$ determination that for any two such elements a and b, $Q_a = Q_b$. The transitive closure on R induces a partition of the set I. We call each member of that partition a "continent". For any x and y that belong to two distinct continents, P_x and P_y should be mutually exclusive, and hence each distinct continent must have at least one distinct nationality. Hence, $|continents| \le |N| \le c$.

From the definition of the $\succ$ determination, it can be seen that each member of a single continent has to be mapped to the same subset of L. Hence, the total number of possible functions is bound from the above by the number of ways the continents can be mapped to subsets of L. Since the number of subsets of L is bounded by 2^l, the total number of functions is bounded by $(2^l)^{|continents|}$. Since $|continents| \le |N| \le c$, the number of functions

Function BuildContinentMap (**input:** ϵ, δ, n);
Collect $\frac{1}{\epsilon}(c(n)l(n)\ln 2 + \ln(\frac{1}{\delta}))$ examples in S
$T := \{\}$
For each example $(\langle x, P_x\rangle, Q_x) \in S$ do
    If $\exists\, T(w)$ such that $w \bigcap P_x \neq \{\}$
        Then Remove $T(w)$ from T and let $T(w \bigcup P_x) := Q_x$
        Else let $T(P_x) := Q_x$
End;
Output T
End BuildContinentMap;

Figure 4: A Learning algorithm for $F_\succ$

in F_n is bounded by $(2^l)^c$. Hence, the dimension of $F_\succ \leq cl$. If c and l are polynomials in n, then by Theorem 1, $F_\succ$ is feasibly learnable since it is of polynomial dimension.

BuildContinentMap (see Figure 4) takes the error parameters ϵ, δ, and problem size n as inputs, collects a large enough set of examples S, and constructs a mapping T, which is consistent with the examples in S, from subsets of N that correspond to continents to subsets of L. T is a representation of a function in $F_\succ$. If the set of P values of an individual x of the current example intersects with some continent w in the mapping T, then the continent w is merged with P_x, and its image is stored as Q_x. (If all the examples are consistent with the determination, this new image will be the same as the old image.) If there is no such intersection, then T maps P_x to Q_x.

To find out the mapping of a new example, say $\langle w, P_w\rangle$, under the learned function, find any $T(y)$, where y has a non-empty intersection with P_w. If the examples are consistent with the determination and with each other, all such y's should give the same result.

T is consistent with the training examples if some function in $F_\succ$ is consistent with them. Note that the size of the table T never exceeds c because the number of entries in T is not increased unless the algorithm comes across an example with nationalities (in P_x) which have not been seen previously. Hence, BuildContinentMap runs in time polynomial in the sample size and c, which are in turn polynomials in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and n. Since the construction of T is consistent with the training examples and the determination $P(x, y) \succ Q(x, z)$, by Theorems 2 and 3, it follows that $F_\succ$ is polynomial-time learnable by BuildContinentMap $\bullet$

Note that BuildContinentMap assumes that the polynomial bounds $c(n)$ and $l(n)$ are known in advance to facilitate the estimation of the number of examples sufficient for learning. Hence, strictly speaking, our proof of Theorem 4 is only an existence proof of a learning algorithm. However, it is possible to convert it into an "on-line" learning algorithm that does not assume the knowledge of these bounds by using the stochastic testing method introduced in [Angluin, 1988]. This method works by incrementally training the system until it correctly classifies a set of randomly drawn test examples. The number of test examples in each iteration must be increased by a small factor to guarantee that the total probability of learning a function which is not approximately correct in any iteration is bounded by δ. If $c(n)$ and $l(n)$ are polynomial functions of n, then the total number of training and test examples can still be shown to be polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and n. We ignore this refinement in the interest of simplicity and simply note that the same argument applies to all of our algorithms, which assume the knowledge of various fixed polynomial functions.

As we mentioned earlier, there are several types of total determinations. We now introduce the remaining types, and characterize when they define learnable function spaces.

In the real world, the assertion that $Nationality(x, y) \succ Language(x, z)$ is too strong. In our toy example, if Giuseppe happens to be a national of both US and Italy, then it follows that Lisa, John, Giuseppe, and Isabella should all speak both Italian and English! This conclusion follows even though Isabella and John do not share any nationality. One individual who has multiple nationalities can effectively merge all those nationalities into a single continent! The following total determination eliminates this problem by weakening the $\succ$ determination.

Definition 11 *Let P and Q be any two binary predicates. We say $P(x, y) \succ_\forall Q(x, z)$ iff*

$$\forall w, x[[\forall y P(w, y) \Leftrightarrow P(x, y)] \Rightarrow \forall z[Q(w, z) \Leftrightarrow Q(x, z)]]$$

$Nationality(x, y) \succ_\forall Language(x, z)$ means that two individuals speak the same set of languages if their set of nationalities is the same. The following theorem characterizes the learnability of $F_{\succ_\forall}$.

Theorem 5 *The space of functions $F_{\succ_\forall}$ consistent with $P(x, y) \succ_\forall Q(x, z)$ is not learnable if $|range(P)| = c \geq O(n)$, where $n = |x|$. However, $F_{\succ_\forall}$ is polynomial-time learnable if $c \leq O(\log n)$.*

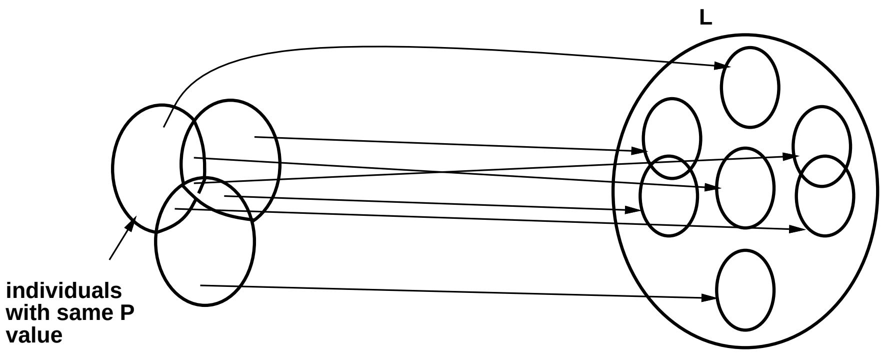

Figure 5: Part of a function in the space consistent with the $\succ_\forall$ determination

Proof: Figure 5 illustrates a portion of a function in the space $F_{\succ_\forall}$. In contrast to the situation in Figure 3, here every subset of N can be mapped to a completely distinct subset of L. Note that elements merely sharing a P value may not be mapped to the same subset of L, as the figure illustrates.

Function BuildSetsOfCountriesMap (**input:** ϵ, δ, n);
Collect $\frac{1}{\epsilon}$ $\{2^{c(n)}l(n)\ln 2 + \ln\frac{1}{\delta}\}$ examples in S
$T := \{\}$
For each example $(\langle x, P_x\rangle, Q_x)$ in S do
    If there exists no entry for P_x in the table T
        Then let $T(P_x) := Q_x$;
        ;; Else $T(P_x)$ must already equal Q_x if the examples are consistent.
End;
Output T
End BuildSetsOfCountriesMap;

Figure 6: A Learning algorithm for $F_{\succ_\forall}$ and $F_{\succ_\subseteq}$

In this case, given any two elements a and b in I, we can assert that $Q_a = Q_b$ if $P_a = P_b$. If $c \leq n$, then $|F_{\succ_\forall}|$ is as large as the number of ways in which subsets of N can be assigned elements in 2^L. This is so because every possible assignment of elements in 2^L to subsets of N defines a function consistent with the above determination. So, in the worst case, $|F_{\succ_\forall}|$ $= (2^l)^{2^c}$. Thus $dim(F_{\succ_\forall}) = 2^c l$. If $c > n$, then $dim(F_{\succ_\forall}) = dim(F) = 2^n l$. This is because $|F_{\succ_\forall}| \leq |F|$, since $F_{\succ_\forall} \subseteq F$. Thus, if $c \geq O(n)$, it follows from Theorem 1 that $F_{\succ_\forall}$ is not feasibly learnable. If $c \leq O(\log n)$, then $dim(F_{\succ_\forall}) = 2^c l \leq 2^{k\log n} l = n^k l$ is a polynomial in n, and hence $F_{\succ_\forall}$ is feasibly learnable.

We describe, in Figure 6, a polynomial-time learning algorithm that collects and identifies a large enough set of examples in time polynomial in n when $c \leq O(\log n)$. It constructs a table T that represents a mapping from subsets of N that represent P_x to subsets of L that represent Q_x for all examples $(\langle x, P_x\rangle, Q_x)$. Since all individuals x who map exactly to the same P_x, should also map to the same Q_x, this table T is guaranteed to be consistent with all the examples. Using T, any input $\langle x, P_x\rangle$ is mapped to $T(P_x)$, or to $\{\}$ if $T(P_x)$ is not in T.

The size of the table T can grow as big as the number of subsets of N. Hence the complexity of the above algorithm is $O(\frac{1}{\epsilon}\{2^c l + \ln\frac{1}{\delta}\})$. If c grows at most logarithmically with n, BuildSetsOfCountriesMap runs in time polynomial with n •

We introduce the remaining types of total determinations below, but postpone their learnability analysis to Appendix 8.1.

One problem that the $\succ_\forall$ determination does not solve is when two individuals, such as Guiseppe and John, *share* a nationality. In that case, we feel confident in asserting that they share a language too. This is captured by the following third type of total determination:

Definition 12 *Let P and Q be any two binary predicates. We say $P(x,y) \succ_\exists Q(x,z)$ iff*

$$\forall w, x[[\exists y\ P(w,y) \wedge P(x,y)] \Rightarrow \exists z[Q(w,z) \wedge Q(x,z)]]$$

In this case $Nationality(x,y) \succ_\exists Language(x,z)$ means that if two individuals share a nationality, then it can be asserted that they share a language. We prove the following negative result in the appendix.

Theorem 6 *The space of functions $F_{\succ_\exists}$ consistent with $P(x,y) \succ_\exists Q(x,z)$ is not learnable.*

For the case when individuals with multiple nationalities exist, it would be computationally advantageous if we could compute the set of languages of such individuals as the union of some "official" set of languages associated with each nationality. Denoting such a relation from nationalities to languages by a second order predicate R, we have the following definition:

Definition 13 *Let P and Q be any two binary predicates. We say $P(x,y) \succ_R Q(x,z)$ iff*

$$\exists R\ \forall x, z[Q(x,z) \Leftrightarrow \exists y[P(x,y) \wedge R(y,z)]]$$

Thus given any nationality y, the set of languages associated with y is simply $\{z \mid R(y,z)\}$. The set of languages spoken by any individual x is given by $\{z \mid P(x,y) \wedge R(y,z)\}$. We prove the following theorem in the appendix.

Theorem 7 *The space of functions $F_{\succ_R}$ consistent with a determination $P(x,y) \succ_R Q(x,z)$ is polynomial-time learnable if $|range(P)| = c$ and $|range(Q)| = l$ are polynomials in $|x| = n$.*

The definition of $P(x,y) \succ_R Q(x,z)$ above is expressed as a statement in second order logic. Russell introduced another determination which is intended to be a first order approximation of the previous determination.

Definition 14 *Let P and Q be any two binary predicates. We say $P(x,y) \succ_\subseteq Q(x,z)$ iff*

$$\forall w, x[[\forall y P(w,y) \Rightarrow P(x,y)] \Rightarrow \forall z[Q(w,z) \Rightarrow Q(x,z)]]$$

In terms of the nationality example, the $\succ_\subseteq$ determination states that if the set of nationalities of w is a subset of that of x, then the set of languages spoken by w is also a subset of those spoken by x. In the appendix, we show that if $c \leq O(\log n)$, then the function class defined by the above determination is learnable in polynomial-time by the algorithm BuildSetsOfCountriesMap in Figure 6. Somewhat surprisingly, however, this function class is not learnable when $c \geq O(n)$, even though the previous function class $F_{\succ_R}$ it is intended to approximate is learnable if $c \leq O(n^k)$.

Theorem 8 *The space of functions $F_{\succ_\subseteq}$ consistent with $P(x,y) \succ_\subseteq Q(x,z)$ is not learnable if $|range(P)| = c \geq O(n)$, where $|x| = n$. However, $F_{\succ_\subseteq}$ is polynomial-time learnable if $c \leq O(\log n)$, and $|range(Q)| = l \leq O(n^k)$.*

4.4 Results on Extended and Partial Determinations

Preliminaries More often than not, real world knowledge admits exceptions. Extended and partial determinations are two types of determination knowledge that can deal with exceptions. In order to facilitate the analysis of such determination knowledge, we introduce a distance metric on function spaces. We then prove a general result regarding the learnability of function spaces that are "close" to other learnable function spaces in terms of this metric.

We begin by defining the notion of *distance* between two functions.[2]

Definition 15 *Given any two functions $f : D \rightarrow R$ and $g : D \rightarrow R$, the* **distance** *between f and g is defined as:*

$$dist(f, g) = |\{x \in D \mid f(x) \neq g(x)\}| \tag{1}$$

In other words, the distance between two functions is simply the number of domain elements on which the two functions disagree. We generalize this notion to function spaces as follows. Intuitively, the distance from a function space to another is the maximum of the distances from the functions in the first space to their closest neighbors in the second space.

Definition 16 *Given any two function subspaces F_n and G_n the* **distance** *from F_n to G_n is defined as:*

$$Dist(F_n, G_n) = Max_{f \in F_n}\{Min_{g \in G_n} dist(f, g)\} \tag{2}$$

Note that the distance from a function subspace F_n to G_n is not necessarily the same as the distance from G_n to F_n. We now define a relation "p-close" between two function spaces that indicates that the distance between the corresponding subspaces is small. It is easy to see that the relation p-close is not symmetric.

Definition 17 *Given two spaces of functions F and G, we say that F is* **$p(n)$-close** *or* **p-close** *to G iff for all n, $Dist(F_n, G_n) \leq p(n)$.*

This relation between function spaces establishes a relationship between their dimensions, which in turn relates the number of examples needed to learn them.

Theorem 9 *Let F be a function space which is $p(n)$-close to G. Let the range of the functions in the two subspaces F_n and G_n be R_n. If $|R_n| \leq 2^{k(n)}$ for some polynomial $k(n)$, and the dimension of G is $Dim_G(n)$, then the dimension of F $Dim_F(n) \leq Dim_G(n) + p(n)k(n) + np(n) + \log p(n)$.*

Proof: Let the domain of the functions in the function subspaces F_n and G_n be D_n. As before, assume that $|D_n| = 2^n$. Since F is p-close to G, $Dist(F_n, G_n) \leq p(n)$. We define a new function subspace $E_n : D_n \rightarrow R_n \cup \{\perp\}$, where $\perp$ is some element not in R_n, as follows.

$E_n = \{f : D_n \rightarrow R_n \cup \{\perp\} \mid f$ maps at most $p(n)$ elements in D_n to R_n, and the rest to $\perp\}$.

[2] The definitions below assume that D, R, F_n, and G_n are all finite.

Intuitively, functions in E_n represent the set of all possible ways in which the functions in F_n can differ with the functions in G_n. Note that

$$|E_n| = \sum_{i=0}^{p(n)} \binom{2^n}{i} (2^{k(n)})^i \leq p(n)(2^n)^{p(n)}(2^{k(n)})^{p(n)}$$

Hence the dimension of E,

$$Dim_E(n) \leq \log p(n) + np(n) + p(n)k(n)$$

Given the spaces G_n and E_n, we define the *product* space $G_n \times E_n$ as follows:

$$G_n \times E_n = \{ f : D_n \rightarrow R_n \mid \exists g \in G_n \text{ and } e \in E_n \text{ such that } \\ \forall x \in I, \text{ if } e(x) = \perp \text{ then } f(x) = g(x), \text{ else } f(x) = e(x) \ \}$$

Since the functions in F_n have a corresponding function in G_n which differs from it on at most $p(n)$ elements, and some function in E_n represents all such differences, it follows that $G_n \times E_n$ includes all functions in F_n. $G_n \times E_n$ contains at most $|G_n| * |E_n|$ many functions.

Hence, the dimension of F,

$$Dim_F(n) = \log |F_n| \leq \log |G_n| + \log |E_n| \leq Dim_G(n) + p(n)k(n) + p(n)n + \log p(n) \bullet$$

We are now ready to infer the feasibility of learning in one space from the feasibility of learning in another space which is $p(n)$-close to it.

Theorem 10 *If a space of functions F is $p(n)$-close to another space of functions G for some polynomial function $p(n)$, and G is feasibly learnable, then F is also feasibly learnable.*

Proof: Since G is feasibly learnable, from Theorem 1, its dimension $Dim_G(n)$ is a polynomial in n. From the previous theorem, it follows that F has a polynomial dimension as well, which implies that F is feasibly learnable $\bullet$

Note that the above theorems do not make any guarantees about the learning time. However, they are useful to predict the number of examples sufficient to learn a function space from the number of examples sufficient to learn another function space which is "close" to it.

Extended Determinations In many real world situations, it is difficult to find total determinations. Even in our nationality example, the reader might have noticed that it is not always true that all people with the same nationality speak the same set of languages. One would like to be able to tolerate a small number of "exceptions" to a total determination. Russell proposes two solutions to this problem: *extended determinations*, and *partial determinations* [Russell, 1986]. We analyze the former first.

An extended determination is like a total determination, except that one is required to see p examples with the same values for P and Q, in order to conclude that all elements with the same value for P also have the same value for Q. An extended determination reduces to a total determination when $p = 1$. Intuitively, extended determinations represent situations where there are a small number of exceptions to a total determination.

Figure 7 illustrates a portion of a function consistent with an extended determination. A set of elements sharing a given P value all map to a given subset of L, except for a set of "exceptional" individuals who map to larger subsets of L.

We now carry out an analysis of extended determinations. Following [Russell, 1986] we define the notion of an *extended determination* as follows:

Definition 18 *We say $P(x, y) \succ_E^p Q(x, z)$ iff*

$$\forall w_1, \ldots, w_p, y, z[P(w_1, y) \wedge Q(w_1, z) \wedge \ldots \wedge P(w_p, y) \wedge Q(w_p, z) \wedge w_1 \neq w_2 \wedge w_1 \neq w_3 \wedge \ldots \wedge w_{p-1} \neq w_p] \Rightarrow \forall x[P(x, y) \Rightarrow Q(x, z)]]$$

For example, $Nationality(x, y) \succ_E^p Language(x, z)$ means if p or more distinct people are American, and they all speak English, then every American will speak English. Clearly, when $p = 1$, this reduces to the determination $\succ$, and as p grows larger the statement becomes weaker. The question naturally arises: how large can p get without sacrificing learnability. To answer this question, we first relate p to the distance between the function subspaces of $F_{\succ_E^p}$ and $F_{\succ_R}$.

Lemma 1 *Let $\mathcal{P} \subseteq I \times N$ and $\mathcal{Q} \subseteq I \times L$. If $|L| \leq l$, and $|N| \leq c$ then the function space $F_{\succ_E^p}$ consistent with the determination $P(x, y) \succ_E^p Q(x, z)$ is $cl(p-1)$-close to the function space $F_{\succ_R}$ consistent with $P(x, y) \succ_R Q(x, z)$.*

Proof: Please see Appendix 8.2.

We are now ready to establish the learnability of the corresponding function space.

Theorem 11 *The function space $F_{\succ_E^p}$ consistent with the determination $P(x, y) \succ_E^p Q(x, z)$ is polynomial-time learnable if $|range(P)| = c$, $|range(Q)| = l$, and p are polynomials in $|x| = n$.*

Proof: The feasible learnability of $F_{\succ_E^p}$ follows from Theorem 10, and the fact that $F_{\succ_E^p}$ is $cl(p-1)$-close to $F_{\succ_R}$, which is feasibly learnable. Please see Appendix 8.2 for a polynomial-time learning algorithm •

Our learnability result essentially states that as long as the number of exceptions permitted by the extended determination is low (that is, polynomial in n), the function space defined by it is polynomial-time learnable.

Partial Determinations A partial determination is similar to an extended determination in that it tolerates a small number of exceptions to a total determination. A partial determination is introduced through a probability measure, $d(P, Q)$, which is an empirical estimate of the relevance of one attribute P to another attribute Q [Russell, 1986]. This measure is also similar to the "uniformity" measure discussed in [Davies, 1988].

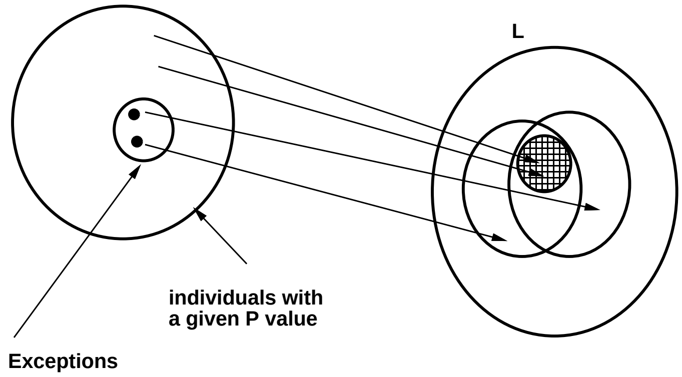

Figure 7: Part of a function in the space consistent with an extended determination

Consider two relations P and Q as before, where $P \subseteq I \times N$ and $Q \subseteq I \times L$. Let us denote the set $\{x \mid P(x,w)\}$ by P_w^{-1}. For each w such that $|P_w^{-1}| > 1$, $d(P_w^{-1}, Q)$ is defined as follows:

$$d(P_w^{-1}, Q) = \left(\frac{1}{|P_w^{-1}|\,(|P_w^{-1}| - 1)} \right) \sum_{i \in P_w^{-1}} \sum_{j \in P_w^{-1} - i} \frac{|Q_i \cap Q_j|}{|Q_j|} \tag{3}$$

If we interpret P as *Nationality* and Q as *Language*, then essentially $d(P_w^{-1}, Q)$ is measuring the extent to which the languages spoken by individuals belonging to nationality w overlap. If they all speak the same set of languages, $d(P_w^{-1}, Q)$ is 1, and if there is no overlap, then $d(P_w^{-1}, Q)$ is 0.

Now we define $d(P,Q)$ as the average of the above metric over all possible range values of the relation P.

$$d(P,Q) = \frac{\sum_{w \in N \text{ and } |P_w^{-1}| > 1} d(P_w^{-1}, Q)}{|\{w \in N : |P_w^{-1}| > 1\}|} \tag{4}$$

In general, $d(P,Q)$ is intended to capture the probability that two randomly chosen individuals with the same P value have the same Q value. Note that if we have a total determination $P \succ Q$, $d(P,Q) = 1$. If P and Q are *uncorrelated*, then $d(P,Q)$ is just the probability that two randomly chosen individuals have the same Q value. Thus, intuitively $d(P,Q)$ is a measure on how relevant P is to making predictions about Q. A good discussion of how this measure is related to other metrics for relevance in statistics, such as correlation, is given in [Davies, 1988]. desJardins considered the task of predicting the value of a target feature Q from a single input feature P, which is assumed to be statistically correlated to Q, making some strong assumptions on the distribution of the input and output features [desJardins, 1989]. In the spirit of our previous results, we derive distribution-independent sufficient conditions for learnability which rely on the asymptotic growth rate of $d(P,Q)$ with the size of the learning problem.

Definition 19 *We say that $P(x,y)$* **partially determines** *$Q(x,z)$ (written as $P(x,y) \succ_P^\alpha Q(x,z)$) if there is a polynomial α in n, such that $d(P,Q) \geq 1 - \frac{\alpha(n)}{2^n}$, where $|x| \leq n$.*

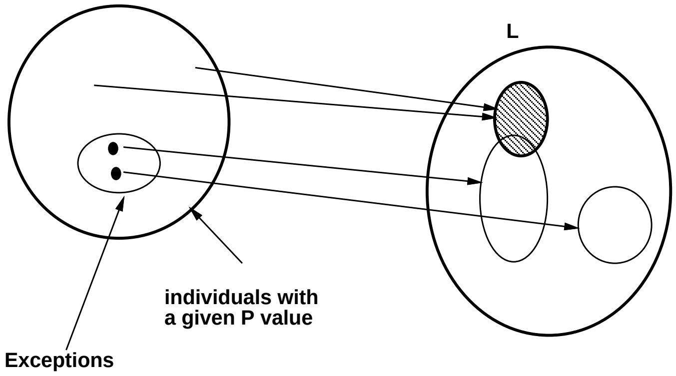

Figure 8: Part of a function in the space consistent with the $\succ_p$ determination

Figure 8 illustrates a portion of a function in the function space $F_{\succ^{\alpha}_{P}}$. Most members of I with a given P value get mapped to a particular subset of L (shown by the shaded region in Figure 8), while there is a set of "exceptions," who get mapped to arbitrary subsets of L. Our results imply essentially that partial determinations define learnable function spaces as long as the number of such exceptions remains bounded by a polynomial in n. Also, note the difference between Figure 7 and Figure 8. In the former, which describes a function consistent with an extended determination, exceptional individuals must map to subsets of L that include the corresponding maps of normal individuals. In the case of partial determinations, there is no such restriction.

If P partially determines Q, then it can be shown that the resulting function space $F_{\succ^{\alpha}_{P}}$ is $2c^2l\alpha$-close to $F_{\succ}$, and hence is feasibly learnable. But first, we need an auxiliary lemma that allows us to infer a lower bound on $d(P_w^{-1}, Q)$ for any $w \in N$ from the lower bound on $d(P, Q)$.

Lemma 2 *If $P \subseteq I \times N$ and $Q \subseteq I \times L$ are such that $P(x,y) \succ^{\alpha}_{P} Q(x, z)$, then for all $w \in N$ s.t. $|P_w^{-1}| > 1$, $d(P_w^{-1}, Q) \geq 1 - \frac{c\alpha}{2^n}$.*

Proof: Please see Appendix 8.3.

Lemma 2 is used to prove the following:

Lemma 3 *The space of functions $F_{\succ^{\alpha}_{P}}$ consistent with $P(x, y) \succ^{\alpha}_{P} Q(x, y)$ is $2c^2l\alpha$-close to the space $F_{\succ}$ consistent with $P(x, y) \succ Q(x, z)$.*

Proof: Please see Appendix 8.3.

Now we can state the main theorem of this section.

Theorem 12 *The space of functions $F_{\succ^{\alpha}_{P}}$ consistent with $P(x, y) \succ^{\alpha}_{P} Q(x, y)$ is polynomial-time learnable, if $|range(P)| = c$, $|range(Q)| = l$, and α are polynomials in $|x| = n$.*

Proof: Feasible learnability follows from the above lemmas and the Theorem 10. Please see Appendix 8.3 for a polynomial-time learning algorithm •

There are some interesting points to note regarding the above analysis. The strategy used to prove the above results, based on a distance metric on function spaces, reveals relations between the various determinations not obvious from their respective definitions. For example, the positive results on both the partial and the extended determinations are due to the proximity of their function spaces to that of a learnable total determination. If we interpret $d(P, Q)$ as the probability of the predicate P totally determining Q, P partially determines Q means that this probability can be made arbitrarily close to 1, by increasing n. Our results show that for learning to be effective, it is not necessary for P to totally determine Q. It suffices if the probability of this can be made asymptotically close to 1.

Interestingly, the notion of partial determination seems similar to the ϵ-semantics, proposed by Pearl to give probabilistic semantics to default logic [Pearl, 1988].[3] Here default rules are interpreted to be sentences which are true with a probability $1 - \epsilon$, where ϵ can be made arbitrarily small. A sentence is true under this semantics if it can be inferred with probability $1 - O(\epsilon)$ in *all* distributions which are consistent with the input defaults. Viewed in this vein, if P partially determines Q, a default inference of a Q value might be sanctioned for an individual with a known P value; however, this default will have to be over-ridden if there is extra-evidence to suggest that the real Q value of this individual is different from that of "normal" individuals.

4.5 A Summary of Learnability Results

Table 1 summarizes our results. The table characterizes the number of examples sufficient to learn the function spaces defined by the different types of determinations under various conditions on their parameters. The table also specifies if and when the function space defined by each determination is polynomial-time learnable.

$Min(x, y)$ denotes the minimum of x and y. Note that in the case of the $\succ_\forall$ determination, since the number of functions consistent with it could not be more than $|F|$, we need to take the smaller of the dimensionalities computed with and without the determination knowledge. In the case of the $\succ_\subseteq$ and $\succ_\exists$ determinations, the dimensionality lies between the two expressions shown enclosed by square brackets.

What can we take away from these results in Table 1?

First, the results characterize some conditions under which the determinations define learnable function spaces. Obviously, the table is not exhaustive in terms of conditions – for example, what happens to the learnability of $F_{\succ_\forall}$, if we further require that every individual can only belong to a constant number of nationalities?[4] The table gives us the sufficient conditions for feasible learning when different determination-based theories are available. It is important, particularly when interpreting the negative results, to note that the results are tied to the assumptions underlying the PAC learnability model. For example, the non-learnable spaces may turn out to be learnable for some specific "easy" distributions of examples.

[3] We thank one of the reviewers, who pointed this out.

[4] As the reader might have guessed, the function space is learnable if the number of nationalities is bounded by a polynomial in n, and not learnable otherwise.

Determ.	*Dimension*	$\lvert Examples \rvert$ *needed*	*P-time learnable if*
$P \succ Q$	$\leq cl$	$\frac{1}{\epsilon}\{cl \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(n^k)$
$P \succ_R Q$	cl	$\frac{1}{\epsilon}\{cl \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(n^k)$
$P \succ_\forall Q$	$Min[2^c l, 2^n l]$	$\frac{1}{\epsilon}\ \{Min(2^c l, 2^n l) \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(\log n)$
$P \succ_\subseteq Q$	$[2^{c/2} l, Min[2^c l, 2^n l]]$	$\frac{1}{\epsilon}\{2^c l \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(\log n)$
$P \succ_\exists Q$	$[2^n (l-1), 2^n l]$	$\frac{1}{\epsilon}\{2^n l \ln 2 + \ln \frac{1}{\delta}\}$	Not Learnable
$P \succ_E^p Q$	$\leq cl + cl^2(p-1) +$ $cln(p-1) + \log(cl(p-1))$	$\frac{1}{\epsilon}\{(cl + cl^2(p-1) + cln(p-1) +$ $\log(cl(p-1))) \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(n^k)$
$P \succ_P^\alpha Q$	$\leq cl + 2c^2 l^2 \alpha +$ $2c^2 l \alpha n + \log 2c^2 l \alpha$	$\frac{1}{\epsilon}\{(cl + 2c^2 l^2 \alpha + 2c^2 l \alpha n +$ $\log 2c^2 l \alpha) \ln 2 + \ln \frac{1}{\delta}\}$	$c \leq O(n^k)$

Table 1: A summary of learnability results. See text for explanation

Second, the results specify the amount the input needs to be abstracted for the function learning problem to be feasible. For example, for the $\succ_R$ determination, a logarithmic reduction is sufficient (in terms of the nationality example, the number of nationalities needs to be a logarithm of the number of people). On the other hand, for the $\succ_\forall$ determination, a doubly logarithmic reduction is needed. Interestingly, for the $\succ_\exists$ determination, even a doubly logarithmic reduction is insufficient to make the learning feasible.

Third, the results also give us some insight into the relations between the various determinations. For example, it is interesting to note that the function space that corresponds to $\succ_R$ is learnable under much weaker conditions than that of its first-order approximation $\succ_\subseteq$.

Finally, the results on extended and partial determinations are interesting because they allow a small number of exceptions to total determinations without sacrificing the learnability. Determinations of this form are more suitable to the real world domains since there are usually a number of exceptions to any rule in such domains.

5 Discussion

The main lesson of this work is that PAC learning can be used to quantify semantic bias by analyzing the learnability of a hypothesis space consistent with *all* prior knowledge – syntactic and semantic. What this work suggests is that in some cases semantic bias can be as effective as syntactic bias in making the learning task tractable. In this section we discuss some implications of our analysis for knowledge-based learning.

5.1 Syntactic and Semantic Biases

Traditionally, PAC learning has focused on constraining learning by providing the learner with a syntactically defined space of functions. For example the class of k-CNF boolean formulas, which is defined as the set of conjunctions of at most k disjuncts, is known to be learnable [Valiant, 1984]. Semantic bias allows one to make finer distinctions between the objects of the same syntactic type. For example, it allows us to talk about something being a function of one attribute rather than another, which cannot be done with purely syntactic constraints. However the price paid is that the semantic bias can be very domain-specific and an algorithm to implement an arbitrary domain-specific bias may be of no use in another domain. The problem, then, is to identify general *classes* of semantic bias more expressive than syntactic bias. Such semantic bias should be describable by a small set of parameters which can be explicit inputs to the learning algorithm. Determinations are good examples of such semantic bias in that they can be parameterized by a set of attributes that determine the target function. This allows the same learning algorithm to be applicable to many domains, in spite of the domain-specific nature of the particular determinations such as nationality determines language.

Note, however, that our work is agnostic about whether prior knowledge should be declaratively represented in the system. The analysis holds whether or not this is the case. In [Russell and Grosof, 1989], the benefits of declarative bias are eloquently argued, while in [Brooks, 1991] the necessity of any declarative representations in reasoning and learning is seriously questioned. The problem is that declarative representations of prior knowledge, while being more flexible, might introduce computational intractability. The question of what part of the knowledge must be declaratively represented and how, must be resolved using considerations of computational complexity in addition to the needed generality and flexibility of learning. In the case of determinations, it suffices to declaratively represent the attributes in the determination. Our algorithms do not require the first order logic representations of the various determinations.

5.2 Tree-structured Bias

Our results suggest that knowledge-based learning is not automatically immune to the observed statistical limitations of inductive learning systems [Dietterich, 1989]. The analysis of the kind performed here can help the designer of a learning system decide whether some existing prior knowledge is adequate, or if more knowledge is needed to make learning feasible. It might also suggest the need for learning from additional sources of information such as "membership queries," when learning from random examples alone is not computationally tractable [Angluin, 1988].

It may not always be realistic to find a small number of relevant attributes that can determine a target function. Russell showed that if the learner has a set of determinations structured as a tree, the number of examples needed to learn a target function is significantly reduced even while the total number of relevant attributes is large [Russell, 1989]. Tree-structured bias consists of a tree of attributes such that each attribute at a node is determined by at most a small constant number (k) of other attributes, which are represented as its children. At the root of the tree is the target attribute, whose value is to be predicted, and at the leaves are the input attributes, whose values are given. The learning problem is made difficult by the fact that the non-root internal nodes that represent "intermediate" attributes are not observable during the training or testing.

Note that it is logically sound to replace the tree of determinations with a single "flat" determination that says that the target attribute is determined by the input attributes. However, Russell showed that the tree structure provides additional bias so that a function class which is not feasibly learnable with the flat determination might still be learnable with the tree of determinations.

It had been shown [Pitt and Warmuth, 1990, Kearns and Valiant, 1989] that learning with tree-structured bias from random examples is computationally as hard as inverting some cryptographic functions such as the RSA cryptosystem, which is believed to be intractable. However, there is an efficient learning algorithm for boolean functions that obey tree-structured bias if the learner is also allowed to query the output of the target function for any arbitrary input, i.e., ask "membership queries" [Tadepalli, 1993]. This algorithm was implemented as a program called TSB and was shown to learn the target functions consistent with a determination tree of a few dozen nodes to almost 100% accuracy with a modest number of examples and queries [Tadepalli, 1993]. A knowledge-free induction program (ID3) using the same data could only achieve a maximum of 75% accuracy. This shows the power of the tree-structured determination knowledge in reducing the number of training examples needed for learning, and the usefulness of membership queries in effectively exploiting this knowledge.

5.3 Learning prior knowledge

We could also consider the possibility of the system interactively learning the necessary prior knowledge, perhaps by asking a domain expert directed questions. In the language domain, for example, it makes sense to ask, "do you know any attribute that determines language?" This is a more useful question to ask than "what language does Giuseppe speak?" because an affirmative answer to the first question greatly reduces the difficulty of the learning problem. Thus, our work can provide guidance as to what questions to ask in a new domain to facilitate further knowledge acquisition.

One could also consider learning prior knowledge from examples and some weaker prior knowledge. For example, under some conditions, "nationality determines language" might be learned by knowing that "some attribute determines language."

However, prior knowledge cannot be considerably weakened without sacrificing learnability. To see this, assume that there is some knowledge K (such as a mapping from individuals to languages) which is not feasibly learnable from prior knowledge P_{weak}, but is feasibly learnable given a stronger piece of prior knowledge P_{strong}. Let us consider the possibility of first learning P_{strong} from P_{weak}, and then learning K. If P_{strong} can be learned feasibly from P_{weak} and examples, then since K can be feasibly learned from P_{strong}, $P_{strong} + K$ can be feasibly learned from P_{weak}. This means that K itself can be feasibly learned from P_{weak}, which we know is impossible. Hence P_{strong} must not be learnable from P_{weak}. A corollary of this is that if something is not learnable from a state of tabula rasa, then any prior knowledge that makes such learning feasible is itself not learnable from a state of tabula rasa.

Another approach for acquiring prior knowledge might be through some other means of learning: by being told, for example. However, the results of [Natarajan and Tadepalli, 1988] show that this approach too suffers from the same information theoretic limitations. Simply stated, the set of all possible functions is too large to be learned from a reasonable number of bits whether these bits are interpreted as examples or domain theory or knowledge or bias.

Thus, we must conclude that although some forms of prior knowledge might be learnable, not all forms are, and that what can be feasibly learned is, indeed, limited.

5.4 Application to knowledge-based learning systems

Our work provides a way to analyze the convergence of techniques for completing partial domain theories. For example, PED is a technique that extends EBL to partial domain theories containing determinations by using classified training instances to extract implicative rules from the determinations [Mahadevan, 1989]. PED relies on the fact that if instances $P(A, B)$ and $Q(A, C)$ of a determination $P(x, y) \succ Q(x, z)$ can be proven from a training instance, then the definition of a determination sanctions adding the rule $P(x, B) \Rightarrow Q(x, C)$ to the domain theory. Our results can be used to distinguish situations when PED can feasibly complete a partial theory from those when it cannot.

For example, if the determinations in a domain theory are all of the first type (that is, $P \succ Q$), and instances of the predicates in a determination can always be derived from a given training instance, and $|range(P)|$ and $|range(Q)|$ are polynomial in n for every determination, then our results imply that PED can fill in the gaps in the domain theory from a polynomial number of training instances.

5.5 Application to Speedup Learning

Interestingly, the analysis of the kind performed above can also be carried out for speedup learning systems like the EBL systems. In these systems, the problem is to compute an operational (efficient) specialization of an intractable domain theory from examples. Since the original domain theory of the system is "complete" in the sense that the examples, and the final result of learning are deductive consequences of the domain theory, it might seem that they require a radically different kind of analysis. However, as we show below, a simple re-interpretation of the task of these systems suffices to allow exactly the same kind of analysis.

Suppose that an EBL system is given examples by a teacher, who generates solutions using an operational specialization of a domain theory which is tuned to the training distribution of problems. Before the learning begins, what does the learner know about this operational form of the domain theory in the teacher's mind? Only that it should be entailed by its domain theory. In other words, it has *no* knowledge of the particular operational version of the domain theory, although it knows the domain theory! The examples provide the learner new information about *the teacher's knowledge*, even if they do not provide new information about the domain. In other words, although the domain theory may be "complete" in the sense that any output of the learner must be sound with respect to the initial domain theory, it is too weak to predict *which* of the many possible deductive consequences of the domain theory the teacher has in mind. We can view this situation as learning a particular operational version of the domain theory, which the teacher is trying to communicate to the learner through examples.

Thus, we can interpret the different operational versions of the domain theory that a speedup learning system might potentially learn as its "hypotheses" about the teacher's knowledge. However, only one of these possible hypotheses (functions) is correct, and the task is to find an approximation to the correct function with a high probability.

In [Tadepalli, 1991], an analysis similar to that we used in this paper was carried out for EBL systems when the hypotheses are in the form of sets of macro-operators. The learning algorithm exploits a piece of domain knowledge called "serial decomposability" [Korf, 1985]. This property is relevant to problem solving domains whose states can be represented as discrete valued feature vectors. A domain is said to be serially decomposable when its features can be ordered such that the effect of any operator (and hence a macro-operator) on a feature is a function of, (or is determined by), only that feature and the features that come before it in the ordering [Korf, 1985]. Given such an ordering of features and an operator sequence that achieves the goal values for those features in that order, it is possible to efficiently decompose the operator sequence into general macro-operators by ignoring all the irrelevant features in the initial state. Thus, similar to determinations, the knowledge of serial decomposability allows the learner to effectively generalize from a small number of examples. Although serial decomposability can be detected by exhaustive search, this problem is known to be NP-hard [Bylander, 1992]. Hence the knowledge of the ordering of features that makes a domain serially decomposable also reduces the time complexity of learning.

6 Conclusions

This paper employed the PAC learning framework to analyze the effectiveness of various forms of prior knowledge in learning. We showed that it is possible to use PAC learning to analyze the effectiveness of semantic prior knowledge as well as syntactic prior knowledge, by viewing the prior knowledge as constraining the function space to that which is consistent with it. We used this approach to analyze various forms of determination knowledge. The analysis revealed surprising differences and similarities between the different kinds of determinations. While some kind of determinations make polynomial-time learning possible, some forms of determinations still require exponential number of examples. The analysis also shows similarities between two seemingly different determinations: partial and extended determinations.

Our work describes one way to do a representation-independent quantified analysis of knowledge. We believe that this kind of analysis can also be carried out for systems that learn from intractable and inconsistent domain theories, and constitutes an interesting direction for future work.

By applying the tools of computational learning theory to a more knowledge-intensive form of learning than it is usually applied to, our work shows how PAC-learning can form a theoretical basis for a unified view of learning. Analyses of this kind might also help us understand the structure of various kinds of knowledge and its relationship to the structure of the complexity classes.

7 Acknowledgements

We are indebted to Balas Natarajan for his generous assistance in our work. In addition, we thank the following people for their invaluable comments on earlier versions of this paper: Tom Amoth, Oren Etzioni, Benjamin Grosof, Steve Minton, Tom Mitchell, and Stuart Russell. We thank Alka Indurkhya for helping us with the proofs. Finally, we thank the reviewers for their detailed and helpful comments. The second author is supported by the National Science Foundation under grant number IRI:9111231.

8 Appendix

In what follows, we make the usual assumptions on the bounds of various sets: $|I| \leq 2^n$, $|N| \leq c$ and $|L| \leq l$.

8.1 Total Determinations

Theorem 6: The space of functions $F_{\succ_\exists}$ consistent with $P(x,y) \succ_\exists Q(x,z)$ is not learnable.

Proof: This determination says that if two individuals share a P value then they also share a Q value. We prove that $F_{\succ_\exists}$ is not learnable by constructing a space F'' of exponential dimension which is consistent with this determination. Let $c = |range(P)| = 1$ and $l = |range(Q)| > 1$. Let i be an element in L such that $\forall x$ in I, $i \in Q_x$. Now for every element b in I we can choose to define the set Q_b in exactly 2^{l-1} ways, since $Q_b = \{i\} \cup L'$ where L' is any subset of L - $\{i\}$. Hence $dim(F'') \geq 2^n(l-1)$, and $F_{\succ_\exists}$ is not learnable even if $c = 1$ •

Theorem 7: The space of functions $F_{\succ_R}$ consistent with a determination $P(x,y) \succ_R Q(x,z)$ is polynomial-time learnable if $|range(P)| = c$ and $|range(Q)| = l$ are polynomials in $|x| = n$.

Proof: $\succ_R$ determination says that there is a relation R from N to L such that Q_x for any $x \in I$ is the set $\{z|P(x,y) \wedge R(y,z)\}$. This implies that, given the relation $\mathcal{P}$, the relation $\mathcal{Q}$ from I to L can be computed from R. Hence, an upper bound on the number of mappings from N to 2^L also gives an upper bound on the number of mappings from I to 2^L. Since $|N| = c$, and $\left|2^L\right| = 2^l$, this upper bound is $(2^l)^c$. Hence $\dim(F_\succ) \leq cl$. If c and l are polynomials in n, then by the dimensionality theorem $F_\succ$ is learnable since it is of at most polynomial dimension.

$F_{\succ_R}$ is in fact polynomial-time learnable since it is polynomial-time identifiable by the program BuildNationalitiesMap in Figure 9. The program returns a table T that represents a mapping from N to subsets of L. T maintains, for each member of N, the largest possible subset of L that it can map to while being consistent with all the previous examples. Initially, each member of P_x has no T-image, and Q_x is made its T-image. Subsequently, a new T-

Function BuildNationalitiesMap (**input:** ϵ, δ, n);
Collect $\frac{1}{\epsilon}(c(n)l(n)\ln 2 + \ln(\frac{1}{\delta}))$ examples in S ;
$T := \{\}$;
For each example $(\langle x, P_x \rangle, Q_x)$ in S do
    For each $P' \in P_x$ do
        If there is an entry $T(P') = z$
            Then let $T(P') := z \bigcap Q_x$
            Else let $T(P') := Q_x$
    End;
End;
Output T
End BuildNationalitiesMap;

Figure 9: A Learning algorithm for $F_{\succ_R}$

image is computed as the intersection of the current T-image and Q_x. It is easy to see that this maintains the required semantics for T and runs in time polynomial in n, l, and c.

T can be viewed as a representation of functions in $F_{\succ_R}$. Any input $\langle x, P_x \rangle$ will be mapped to $\{z | z \in T(y) \text{ and } y \in P_x\}$ •

Theorem 8: The space of functions $F_{\succ_\subseteq}$ consistent with $P(x, y) \succ_\subseteq Q(x, z)$ is not learnable if $|range(P)| = c \geq O(n)$, where $|x| = n$. However, $F_{\succ_\subseteq}$ is polynomial-time learnable if $c \leq O(\log n)$, and $|range(Q)| = l \leq O(n^k)$.

Proof: This determination constrains functions from $D \rightarrow 2^L$ in the following way: given any two elements a and b in I, if $P_a \subseteq P_b$, then it must be that $Q_a \subseteq Q_b$. It follows that if there exist a and b in I such that $P_a = P_b$, then $Q_a = Q_b$. Hence, this determination is stronger than $\succ_\forall$, and hence, is learnable whenever $F_{\succ_\forall}$ is learnable.

We prove that it is not learnable under significantly weaker conditions by constructing a space of functions F' of exponential dimension consistent with the above determination. We construct F' as follows. Assume that $c = 2n$. For any b in I, if P_b is of size greater than $c/2$, then define $Q_b = L$. If P_b is of size less than $c/2$, then define $Q_b = \{\}$. If P_b is of size exactly $c/2$, then let Q_b be any arbitrary subset of L. It is easy to verify that F' is consistent with the above determination. Since there exist at least $2^{c/2}$ subsets of N of size $c/2$, and since we can assign each of them to subsets of L in 2^l ways, the number of functions in F' is at least $(2^l)^{2^{c/2}}$. Hence $dim(F') \geq 2^{c/2} l = 2^n l$.

Next we have to show that when $c \leq O(\log n)$, $F_{\succ_\subseteq}$ is polynomial-time learnable. For this, we simply note that $F_{\succ_\subseteq} \subseteq F_{\succ_\forall}$. Since $F_{\succ_\forall}$ is learnable in polynomial-time by Build-SetsofCountriesMap (cf. Figure 6) when $c \leq O(\log n)$, $F_{\succ_\subseteq}$ is also learnable by the same algorithm.

8.2 Extended Determinations

Lemma 1: *Let $\mathcal{P} \subseteq I \times N$ and $\mathcal{Q} \subseteq I \times L$. If $|L| \leq l$, and $|N| \leq c$ then the function space $F_{\succ_E^p}$ consistent with the determination $P(x,y) \succ_E^p Q(x,z)$ is cl(p-1)-close to the function space $F_{\succ_R}$ consistent with $P(x,y) \succ_R Q(x,z)$.*

Proof: Let f be any function in the n^{th}-subspace of $F_{\succ_E^p}$. We show that there is a function g in the n^{th}-subspace of $F_{\succ_R}$ such that $dist(f,g)$ is at most $cl(p-1)$.

Let P and Q be the predicates corresponding to f. T is a partial function from N to 2^L defined as follows.

$T(y) := \{z \mid$ there are p distinct elements $x_i \in I$ such that $P(x_i, y) \wedge Q(x_i, z)\}$

By the definition of extended determination, if $T(y)$ is defined for some $y \in N$, then for all $x \in I$ such that $P(x,y)$, Q_x must include $T(y)$.

If we define $g(\langle x, P_x \rangle)$ to be $\bigcup_{y \in P_x} T(y)$, it then follows that for any function $f \in F_{\succ_E^p}$, $f(\langle x, P_x \rangle)$ must include $g(\langle x, P_x \rangle)$.

Furthermore, treating T as a representation of the relation $R \subseteq N \times L$ as defined by the determination $F_{\succ_R}$, g can be seen to be a member of $F_{\succ_R}$.

We now estimate $dist(f,g)$. Let us call each x on which f and g disagree, an exception. By the definition of T and $F_{\succ_E^p}$, each (y,z) s.t. $y \in N$ and $z \in L$ can contribute at most $p-1$ exceptions. If more than $p-1$ members of I map to the same $y \in N$ and the same $z \in L$, then $T(y)$ will include z, which will be reflected in g. Hence the number of exceptions, $dist(f,g) \leq cl(p-1)$, which implies that the function space $F_{\succ_E^p}$ is $cl(p-1)$-close to $F_{\succ_R}$ •

Theorem 11: *The function space $F_{\succ_E^p}$ consistent with the determination $P(x,y) \succ_E^p Q(x,z)$ is polynomial-time learnable if $|range(P)| = c$, $|range(Q)| = l$, and p are polynomials in $|x| = n$.*

Proof: By Theorem 9 and Lemma 1, the dimension of $F_{\succ_E^p}$ is at most

$$Dim_{\succ_R}(n) + cl(p-1) * l + cl(p-1) * n + \log(cl(p-1))$$

Since $Dim_{\succ_R}(n)$ is at most cl, by Theorem 3, it can be feasibly learned by identifying a set of

$$\frac{1}{\epsilon}\{(cl + cl^2(p-1) + cln(p-1) + \log(cl(p-1)))\ln 2 + \ln\frac{1}{\delta}\}$$

examples.

The procedure BuildNationalitiesAndExceptionsMap (see Figure 10) collects that many examples, and constructs the mapping T from N to 2^L defined in the previous proof, and also builds a table E of exceptions indexed by nationalities and languages. Whenever the number of exceptions for a nationality y and language z exceeds $p-1$, then $T(y)$ is updated to include z, and the exceptions are removed.

Assuming that c, l, and p are polynomials in n, it can be easily seen that the above program runs in time polynomial in the required parameters. Since the size of exceptions table does not exceed p for any nationality language pair (y,z), T and E represent a function consistent with the determination $P \succ_E^p Q$ and with the examples. Hence, BuildNationalitiesAndExceptionsMap is a polynomial-time learning algorithm for $F_{\succ_E^p}$. The tables T and

Function BuildNationalitiesAndExceptionsMap (**input:** ϵ, δ, n);
Collect $\frac{1}{\epsilon}\{(c(n)l(n) + c(n)l^2(n)(p(n)-1) + c(n)l(n)n(p(n)-1) + \log(c(n)l(n)(p(n)-1)))\ln 2 + \ln\frac{1}{\delta}\}$ examples in S
$T := E := \{\}$
For each example $(\langle x, P_x\rangle, Q_x)$ in S do
    For each y in P_x do
        ;; $T(y) \subseteq Q_x$ if the example set is consistent with the determination.
        For each z in $Q_x - T(y)$ do
            If $|E(y,z)| \geq p-1$, Then
                Let $T(y) := T(y) \bigcup \{z\}$
                $E(y,z) := \{\}$
            Else $E(y,z) := E(y,z) \bigcup (\langle x, P_x\rangle, Q_x)$
        End
    End
End
Output T and E
End BuildNationalitiesAndExceptionsMap

Figure 10: A Learning algorithm for $F_{\succ^p_E}$

E are used to predict Q_x for a given example as follows: first look up the particular example in the exceptions table E (for each $y \in P_x$ and $z \in L$). If it is not in this table, then return the union of all $T(y)$ such that $y \in P_x$ $\bullet$

8.3 Partial Determinations

Lemma 2: *If $P \subseteq I \times N$ and $Q \subseteq I \times L$ are such that $P(x,y) \succ^\alpha_P Q(x,z)$, then for all $w \in N$ s.t. $|P^{-1}_w| > 1$, $d(P^{-1}_w, Q) \geq 1 - \frac{c\alpha}{2^n}$.*

Proof: We prove this result by contradiction. Assume that there is some $w' \in N$ such that $\left|P^{-1}_{w'}\right| > 1$ and $d(P^{-1}_{w'}, Q) < 1 - \frac{c\alpha(n)}{2^n}$. Let

$$\left|\{w \in N : \left|P^{-1}_w\right| > 1\}\right| = c'$$

Note that $c' \leq |N| = c$. Since $d(P^{-1}_w, Q) \leq 1$ for all W, from the definition of $d(P,Q)$, it follows that

$$d(P,Q) < \frac{(1 - \frac{c\alpha(n)}{2^n}) + (c'-1)}{c'} = 1 - \frac{c\alpha(n)}{c'2^n} \tag{5}$$

Since P(x,y) $\succ^\alpha_P$ Q(x, z), we have

$$d(P,Q) \geq 1 - \frac{\alpha(n)}{2^n}$$

Combining the above two inequalities, we get

$$c'\alpha(n) > c\alpha(n)$$

which leads to a contradiction •

Lemma 3: *The space of functions $F_{\succ^{\alpha}_{P}}$ consistent with $P(x,y) \succ^{\alpha}_{P} Q(x,y)$ is $2c^2l\alpha$-close to the space $F_{\succ}$ consistent with $P(x,y) \succ Q(x,z)$.*

Proof: We prove this result by contradiction. Assume that $F_{\succ^{\alpha}_{P}}$ is not $2c^2l\alpha$-close to $F_{\succ}$. Then it follows that there is some function f in $F_{\succ^{\alpha}_{P}}$ such that for all functions g in $F_{\succ}$, $dist(f,g) > 2c^2l\alpha$.

Let us denote the restriction of f and g to the elements of P^{-1}_{w} by f_w and g_w respectively. Let $w' \in N$ be such that $dist(f_{w'}, g_{w'}) \geq dist(f_w, g_w)$, for all $w \in N$. It is easy to see that

$$dist(f,g) \leq \sum_w dist(f_w, g_w) \leq c * dist(f_{w'}, g_{w'})$$

We can combine the above two inequalities to obtain

$$dist(f_{w'}, g_{w'}) > \frac{2c^2l\alpha}{c} = 2cl\alpha$$

We now show that the above is not possible, which implies that

$$\forall f \in F_{\succ^{\alpha}_{P}}, \forall g \in F_{\succ}, \forall w \in N, dist(f_w, g_w) \leq 2cl\alpha \tag{6}$$

Note that for any $g \in F_{\succ}$, $g_{w'}$ should map all the elements x in $P^{-1}_{w'}$ to the same set $L' \subseteq L$. If $f_{w'}$ were to differ from *all* such $g_{w'}$ by more than $2cl\alpha$, then, for any set $L' \subseteq L$, there are more than $2cl\alpha$ elements in $P^{-1}_{w'}$ that are not mapped to L' by f. It then follows that for each $i \in P^{-1}_{w'}$, there are more than $2cl\alpha$ elements $j \in P^{-1}_{w'}$ such that $Q_j = f_{w'}(\langle j, P_j\rangle)$ is different from $Q_i = f_{w'}(\langle i, P_i\rangle)$. At least for half such pairs (i,j), $Q_j \not\subseteq Q_i$. Each such pair contributes at most $1 - \frac{1}{l}$ to the numerator of $d(P^{-1}_{w'}, Q)$. Thus, we get the following upper bound for $d(P^{-1}_{w'}, Q)$.

$$d(P^{-1}_{w'}, Q) \leq \frac{\left|P^{-1}_{w'}\right|(\left|P^{-1}_{w'}\right| - 1) - \left|P^{-1}_{w'}\right| 2cl\alpha\frac{1}{2}\frac{1}{l}}{\left|P^{-1}_{w'}\right|(\left|P^{-1}_{w'}\right| - 1)} = 1 - \frac{c\alpha}{(\left|P^{-1}_{w'}\right| - 1)} < 1 - \frac{c\alpha}{\left|P^{-1}_{w'}\right|} \tag{7}$$

From Lemma 2, we know that

$$d(P^{-1}_{w'}, Q) \geq 1 - \frac{c\alpha}{2^n}.$$

Combining the above two inequalities, we get $\left|P^{-1}_{w'}\right| > 2^n$, which leads to a contradiction. Hence $dist(f_{w'}, g_{w'}) \leq 2cl\alpha$, and $dist(f,g) \leq 2c^2l\alpha$ •

Theorem 12: The space of functions $F_{\succ^{\alpha}_{P}}$ consistent with the partial determination $P(x,y) \succ^{\alpha}_{P} Q(x,y)$ is polynomial-time learnable, if $|range(P)| = c$, $|range(Q)| = l$, and α are polynomials in $|x| = n$.

Proof: Instead of directly learning the set of functions $F_{\succ^{\alpha}_{P}}$, our algorithm learns a superset of functions which are $2c^2l\alpha$-close to $F_{\succ}$.

By Theorem 9 and Lemma 1, the dimension of $F_{\succ_P^\alpha}$ is at most

$$Dim_{\succ}(n) + 2c^2 l\alpha * l + 2c^2 l\alpha * n + \log 2c^2 l\alpha$$

Since $Dim_{\succ}(n)$ is at most cl, by Theorem 3, it can be feasibly learned by identifying a set of

$$\frac{1}{\epsilon}\{(cl + 2c^2 l^2\alpha + 2c^2 l\alpha n + \log 2c^2 l\alpha)\ln 2 + \ln\frac{1}{\delta}\}$$

examples.

FindMinimalExceptionsMap (see Figure 11) collects that many examples, and identifies it in time polynomial in n, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$. The idea of this learning algorithm is to build a table T from N to subsets of L, to represent the "normal" mappings, and store the examples that do not obey this mapping in an exceptions table E.

Note that all the "normal" mappings from $y \in I$ whose P values have some $w \in N$ in common should map to the same subset of L. The algorithm considers each $w \in N$, and builds the Table T, which maps each $w \in N$ to a subset Z of L such that the number of exceptions $\langle x, P_x\rangle$ which do not map to that subset, even while $w \in P_x$, are minimized. This is done by computing $Score(w, Z)$ for each $w \in N$, and each relevant $Z \subseteq L$, which indicates the number of members of P_w^{-1} whose Q values coincide with Z. The $Z \subseteq L$ that maximizes $Score(w, Z)$ is stored in $T(w)$. For each $w \in N$ all members of P_w^{-1} whose Q values do not coincide with $T(w)$ are considered "exceptions" and their Q values are stored in E.

Since the target function satisfies the partial determination $F_{\succ_P^\alpha}$, from Equation 6 in the proof of Lemma 3, the number of actual exceptions for each $w \in N$ is upper-bounded by $2cl\alpha$. Since our algorithm minimizes the number of exceptions in each P_w^{-1} with respect to the examples, the number of exceptions stored in E for each w must also be $\leq 2cl\alpha$. Hence the maximum size of E is upper-bounded to $2c^2 l\alpha$, ensuring that the output function represented by T and E is $2c^2 l\alpha$-close to some function in $F_{\succ}$.

It is easy to see that the above program runs in time polynomial in n, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ if c and l are bounded by polynomials in n. The tables T and E are used to predict Q_x for a given input $\langle x, P_x\rangle$ as follows: First look up E for the particular example. If it is not in this table, then return $T(y)$ for some $y \in P_x$ $\bullet$

Function FindMinimalExceptionsMap (**input:** ϵ, δ, n);
Collect $\frac{1}{\epsilon}\{(c(n)l(n) + 2c^2(n)l^2(n)\alpha(n) + 2c^2(n)l(n)\alpha(n)n +$
$\log 2c^2(n)l(n)\alpha(n))\ln 2 + \ln \frac{1}{\delta}\}$ examples in S
$T := E := N := \{\}$;
$Score(*,*) := 0$;
;; Compute the scores and initialize $T(w)$
For each $(\langle y, P_y\rangle, Q_y) \in S$ do
For each $w \in P_y$ do
$Score(w, Q_y) := Score(w, Q_y) + 1$;
$T(w) := \{\}$;
End;
End;
;; Store T-images as the Q values that maximize their Score
For each $\langle w, Z\rangle$ for which $Score$ is computed, do
If $T(w) = \{\}$ or $Score(w, Z) > Score(w, T(w))$ Then
$T(w) := Z$;
End;
;; Store the exceptions in E
For each $(\langle y, P_y\rangle, Q_y) \in S$ do
For each $w \in P_y$ do
If $T(w) \neq Q_y$ and $E(y)$ is not already stored
Then let $E(y) := Q_y$
End;
End;
Output T and E
End FindMinimalExceptionsMap

Figure 11: A Learning algorithm for $F_{\succ^p_E}$

References

[Angluin, 1988] D. Angluin. Queries and concept learning. *Machine Learning*, 2, 1988.

[Blumer *et al.*, 1989] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[Brooks, 1991] R. Brooks. Intelligence without reason. In *Proceedings of the 12th IJCAI*. Morgan Kaufmann, 1991.

[Bylander, 1992] T. Bylander. Complexity results for serial decomposability. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI)*, pages 729–734. Morgan-Kaufmann, 1992.

[Danyluk, 1989] A. Danyluk. Finding new rules for incomplete theories: Explicit biases for induction with contextual information. In *Proceedings of the Sixth International Machine Learning Workshop*. Morgan-Kaufmann, 1989.

[Davies and Russell, 1987] T. Davies and S. Russell. A logical approach to reasoning by analogy. In *Proceedings of The Tenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1987.

[Davies, 1988] T. Davies. Determination, uniformity, and relevance: Normative criteria for generalization and reasoning by analogy. In D. H. Helman, editor, *Analogical Reasoning*, pages 227–250. Kluwer Academic Publishers, 1988.

[Dejong and Mooney, 1986] G. Dejong and R. Mooney. Explanation-Based Learning: An alternative view. *Machine Learning*, 1(2), 1986.

[desJardins, 1989] M. desJardins. Probabilistic evaluation of bias for learning systems. In *Proceedings of the Sixth International Machine Learning Workshop*, pages 495–499. Morgan-Kaufmann, 1989.

[Dietterich, 1989] T. Dietterich. Limitations on inductive learning. In *Proceedings of the Sixth Machine Learning Workshop*, pages 124–128. Morgan Kaufmann, 1989.

[Hall, 1988] R. Hall. Learning by failing to explain. *Machine Learning*, 3(1), 1988.

[Haussler *et al.*, 1988] D. Haussler, N. Littlestone, and M. Warmuth. Predicting $0, 1$ functions on randomly drawn points. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 100–109, 1988.

[Haussler, 1988] D Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artifical Intelligence*, 36(2), 1988.

[Hirsh, 1989] H Hirsh. *Incremental Version-Space Merging: A General Framework for Concept Learning*. PhD thesis, Stanford University., 1989.

[Kearns and Valiant, 1989] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the* 21^{st} *Annual ACM Symposium on Theory of Computing*, 1989.

[Korf, 1985] R. Korf. Macro-Operators: A Weak Method for Learning. *Artificial Intelligence*, 26(1):35–77, 1985.

[Mahadevan and Tadepalli, 1988] S. Mahadevan and P. Tadepalli. On the tractability of learning from incomplete theories. In *Proceedings of the Fifth International Machine Learning Conference*, pages 235–241. Morgan-Kaufmann, 1988.

[Mahadevan, 1989] S. Mahadevan. Using determinations in EBL: A solution to the incomplete theory problem. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 320–325. Morgan-Kaufmann, 1989.

[Mitchell *et al.*, 1986] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), 1986.

[Mitchell, 1980] T. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, 1980.

[Natarajan and Tadepalli, 1988] B. Natarajan and P. Tadepalli. Two new frameworks for learning. In *Proceedings of the Fifth International Machine Learning Conference*. Morgan-Kaufmann, 1988.

[Natarajan, 1987] B. Natarajan. On learning boolean functions. In *ACM Symposium on Theory of Computing*, 1987.

[Natarajan, 1989] B. Natarajan. On learning sets and functions. *Machine Learning*, 4(1), 1989.

[Natarajan, 1991] B. Natarajan. *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, 1991.

[Pazzani, 1992] M. Pazzani. The Utility of Knowledge in Inductive Learning. *Machine Learning*, 9(1):57–94, 1992.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Pitt and Valiant, 1988] L. Pitt and L. G. Valiant. Computational limits of learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[Pitt and Warmuth, 1990] L. Pitt and M. Warmuth. Prediction preserving reducibility. *Journal of Computer and System Sciences*, 41(3), 1990.

[Rich and Knight, 1991] E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, 1991.

[Rivest, 1987] R. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[Russell and Grosof, 1989] S. Russell and B. Grosof. A sketch of autonomous learning using declarative bias. In P. Brazdil and K. Konolige, editors, *Machine Learning, Meta-Reasoning, and Logics*. Kluwer Academic, 1989.

[Russell, 1986] S. Russell. *Analogical and Inductive Reasoning*. PhD thesis, Stanford University., 1986.

[Russell, 1987] S. Russell. Analogy and single instance generalization. In *Proceedings of the 4th International Conference on Machine Learning*, pages 390–397. Morgan-Kaufmann, 1987.

[Russell, 1988] S. Russell. Tree-structured bias. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI)*, pages 641–645. Morgan-Kaufmann, 1988.

[Russell, 1989] S. Russell. *The use of knowledge in analogy and induction*. Morgan Kaufmann, 1989.

[Tadepalli, 1991] P. Tadepalli. A formalization of explanation-based macro-operator learning. In *Proceedings of the IJCAI*, pages 616–622, 1991.

[Tadepalli, 1993] P. Tadepalli. Learning from queries and examples with tree-structured bias. In *Proceedings of the Tenth International Machine Learning Conference*. Morgan-Kaufmann, 1993.

[Valiant, 1984] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11), 1984.