Relational Concepts and the Fourier Transform: an Empirical Study

Eduardo Pérez[1] and Larry Rendell[2]

[1] Universidad Autónoma de Madrid, E.T.S. Informática,
Madrid E-28049, Spain
e-mail: eduardo.perez@ii.uam.es

[2] University of Illinois, Dept. of Computer Science and Beckman Institute
Urbana, IL 61801, USA
e-mail: rendell@cs.uiuc.edu

Abstract. Lack of domain knowledge may impose primitive data representations. Then, complex (non-linear) relationships among attributes complicate learning, especially for typical learning methods. These methods fail because their bias does not match the complex relational structures relevant to the domain. However, more recent approaches to learning have implemented biases that allow learning of structured, albeit complex, concepts. One of such approaches, based on the Fourier transform of Boolean functions, is studied and compared empirically to others, based on constructing new features or extracting relations from propositional training data. Controlled experiments help to characterized the kinds of concept that allow each approach to outperform the others. This characterization, which implicates parameters of Fourier complexity, other measures of concept difficulty, and the relational structure of the target concepts, is also discussed with respect to difficult real-world domains.

1 Introduction: Motivation and Background

When lack of domain knowledge forces the use of primitive data representations, complex relations among attributes prevail and complicate learning significantly. Then, typical learning methods fail because their similarity-based bias does not match the complex relational structures relevant to the domain. It is true that any learner must be biased in some way [18, 8]. Thus, no learning system can be the best uniformly over all concepts [16], mainly because of the extremely large number of random (unstructured) concepts [1]. However, more recent approaches to learning have implemented biases that allow learning of structured, albeit complex, concepts. One of such approaches is to change the representation of the examples, introducing new terms (features or relations among attributes) to narrow the gap between primitive input representation and target concept [4]. Some machine learning methods, such as Fringe [9] and MRP [10], automate this representation change. Another approach, studied in computational learning theory [2, 6], explicitly introduces a representation based on the Fourier transform of Boolean functions in order to capture complex relations underlying the target concept. This paper empirically compares both approaches and characterizes the kinds of concept that allow each approach to outperform the other. This characterization, which implicates parameters of Fourier complexity, other measures of concept difficulty, and the relational structure of the target concepts, is also discussed with respect to difficult real-world domains.

The rest of this section reviews basic definitions and terminology relevant to the Fourier transform, and illustrates this representation in the context of learning. In Section 2, the learning accuracy of four learning systems is described empirically as a function of Fourier complexity. The results from these experiments are discussed and compared with previous ones regarding Concept Variation, another measure of learning difficulty. Then, Section 3 extends the empirical comparison by considering one algorithm from the computational learning theory literature, which is explicitly designed to learn in terms of the Fourier transform. This algorithm is compared again, in Section 4, to the other learners, but this time using target concepts that cover a wider range of the Fourier complexity spectrum and include combinations of complex relations among attributes. Section 5 discusses algorithm limitations and extensions suggested by the empirical comparisons, and analyzes related work.

To consider the Fourier transform in the context of learning Boolean functions, it is convenient to view every Boolean function f as

$$f : \{0,1\}^n \rightarrow \{+1,-1\} \; . \tag{1}$$

Any such function, or concept, can be expressed as the sign of a linear combination of Fourier terms $\chi_z(x)$, that is,

$$f(x) = sign\left(\sum_{z \in \{0,1\}^n} \hat{f}(z)\chi_z(x)\right) , \tag{2}$$

where $sign(x) = +1$ if $x \geq 0$, and $sign(x) = -1$ otherwise. To simplify the notation, the *sign* function is often left out but implicitly understood when converting real numbers to Boolean values. Each $\hat{f}(z)$ is the coefficient of a Fourier term, and each Fourier term $\chi_z(x)$ is a parity function defined by

$$\chi_z(x) = (-1)^{z_1x_1+z_2x_2+\ldots+z_nx_n} \; . \tag{3}$$

That is, the characteristic vector z of the Fourier term $\chi_z(x)$ is a binary vector whose *active* bits (i.e., bits set to 1) serve to select the corresponding bits (or attribute values) from an example input vector x, and then the parity of the selected input bits is the value of $\chi_z(x)$. The number of active bits in z defines the *degree* of a Fourier term $\chi_z(x)$ or a coefficient $\hat{f}(z)$. The coefficient of the Fourier series of f are real numbers that can be computed by

$$\hat{f}(z) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\chi_z(x) \; . \tag{4}$$

Although each real function has a unique Fourier series, different real functions can be interpreted as the same Boolean function (since only the sign is retained). Hence, learning systems approximating Boolean functions by real functions may choose among several Fourier decompositions for the same Boolean function.

The *spectral norm* of f, denoted by $L(f)$, is defined as the sum of absolute values of all Fourier coefficients of f:

$$L(f) = \sum_{z \in \{0,1\}^n} |\hat{f}(z)| \ . \tag{5}$$

It can be shown that $L(f) \geq 1$, for all f. However, $L(f)$ is the sum of 2^n absolute values of Fourier coefficients, possibly resulting in a large sum unless there are only a few non-zero coefficients, or unless the number of coefficients that are "large" is "small". Some classes of Boolean functions have only a few non-zero Fourier coefficients, and consequently have small spectral norm. It is common terminology to refer to polynomially bounded spectral norms as "small". Having small spectral norm implies learnability (with membership queries and under uniform distribution assumptions) [5].

The spectral norm depends on the absolute value, not the degree, of the non-zero coefficients. Consider, for instance, the following functions, each with a single non-zero coefficient of degree 1, 2, 3 and 4 respectively:

$$\begin{array}{ll} p_1(x) = -1 \cdot \chi_{1000\ldots0}(x) & [\equiv x_1] \ , \\ p_2(x) = -1 \cdot \chi_{1100\ldots0}(x) & [\equiv x_1 \oplus x_2] \ , \\ p_3(x) = -1 \cdot \chi_{1110\ldots0}(x) & [\equiv x_1 \oplus x_2 \oplus x_3] \ , \\ p_4(x) = -1 \cdot \chi_{1111\ldots0}(x) & [\equiv x_1 \oplus x_2 \oplus x_3 \oplus x_4] \ . \end{array} \tag{6}$$

They all have spectral norm equal to 1, but may not be equally difficult to learn. In particular, for learning methods based on the similarity-based bias [14], the first function is a trivial target concept; the others are typically considered of highest difficulty. Nevertheless, the small spectral norm of these and other *hard* concepts suggests that an algorithm based on the Fourier transform can learn them under certain circumstances (e.g., with the help of membership queries).

The set of Fourier terms constitutes a basis that allows any function to be expressed as a linear combination of the basis (parity) functions. Thus, non-linear relationships among variables are captured *within* the Fourier terms, not by the Fourier series itself. That is, complex attribute interaction can be expressed as simple (linear) combinations of complex basis functions. This is also a goal of some machine learning methods considered in the next section.

2 Fourier Complexity and Learning Performance

This section empiricallly studies learning performance as a function of Fourier complexity. The complexity of a Fourier series can be characterized by two parameters: the number and the degree of Fourier terms with non-zero coefficients. The following experiments show the effect of these parameters on the learning accuracy of several learning systems. The systems considered here are C4.5

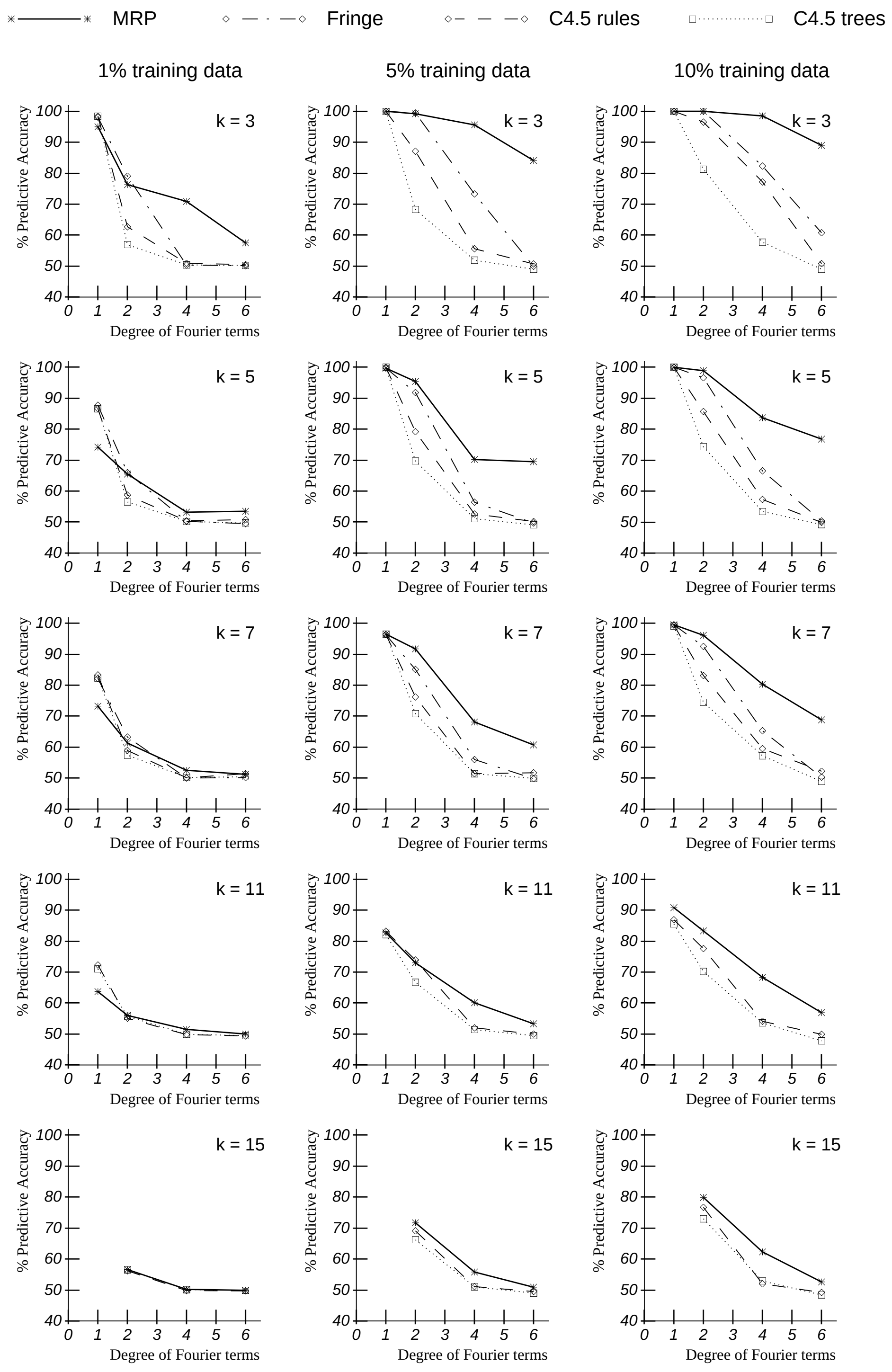

Fig. 1. Predictive accuracy as a function of Fourier complexity

trees, C4.5 rules, Fringe, and MRP. C4.5 provides the baseline performance of a similarity-based learner that does not change the representation of the examples during learning [13]. Fringe, however, changes the representation by introducing new features that result from combining attribute pairs at the end of each path from the root of a decision tree constructed using a previous representation [9]. MRP, which finds relations among primitive attributes by means of multidimensional projection, extracts their extensional representation from the training data, and uses them as new predicates for the test nodes of a decision tree [10].

The Boolean functions used here as target concepts are characterized by two parameters: they have at most k non-zero Fourier terms, and each term has degree d. For each pair of values (k, d), 8 functions of 12 variables were generated by randomly choosing their Fourier coefficients from $\{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$. The performance measure used was predictive accuracy (i.e., percentage of correct classification measured over all data points not used for training). Each algorithm was run 5 times for each function and size of training data, except that Fringe was too slow to be run for $k > 7$. For each algorithm, its average accuracy for different values of k, d, and training size are shown in Figure 1. Each of these averages summarizes 40 accuracy values (5 values for each of 8 target concepts). The number of terms k was set to values 3, 5, 7, 11, and 15 to observe its effect in learning difficulty. The three graphs in each row of Figure 1 correspond to a particular value of k. Each column corresponds to a different size of training data. Each graph plots an algorithm's accuracy as a function of d, the degree of the Fourier terms (or number of parity bits). Note that $d = 1$ is the extreme case of no parity involved, and so the functions generated with $d = 1$ are weighted linear thresholds of at most k input variables.

Figure 1 shows an expected degradation of performance for the four learning systems as either parameter, k or d, of the Fourier series increases. However, d seems to have a stronger effect than k. Consider the leftmost column of graphs. Even for a very small number of terms (e.g., 3), $d = 6$ is sufficient to bring all learning curves to the level of guessing, and thus increasing k cannot make things worse once d is high (≥ 6). When d is low (e.g., 2), then increasing the number of terms reduces accuracy notably. However, this effect is more gradual (for the k values used here). Note that d is bounded by n, but k can be as large as $n(n-1)\ldots(n-d+1)$. Since the spectral norm (i.e., the sum of absolute values of the Fourier coefficients) does not depend on d, it seems to capture only a certain aspect of learning difficulty: *one which is independent of the degree of parity involved*. However, most current learning systems are extremely sensitive to (even small) increases in the degree of the Fourier terms.

The performance of MRP (relative to that of the other learners) in this context is similar to what was observed in the context of increasing concept variation [11]. Here, MRP's degradation with increasing d (degree of parity) is not worse and sometimes considerably better than the degradation experienced by the other learners. Also, as training data increases (from left to right column in Figure 1), MRP is the only one of the four systems studied that can take advantage of increasing data regardless of function complexity. Nevertheless, when the number of Fourier terms becomes large (e.g., 15), a training sample of 1% is not sufficient to allow learning by any of the systems used here.

The leftmost column of graphs in Figure 1 also shows that MRP's advantage has a counterpart. When $d = 1$, no complex interactions are involved; the target functions are linear thresholds. The other learners can find a more accurate hypothesis because their SBL bias is appropriate for these concepts. MRP's less restrictive bias is more likely to overfit spurious regularities appearing in small data samples. On the other hand, larger data samples remove this disadvantage (e.g., consider the accuracies for $d = 1$ in graphs $k = 7$ and 11 as data increases).

The above analysis, in particular with respect to the effect of the degree of Fourier terms in performance, suggests a link between Fourier complexity and Variation (used in [11] as a measure of concept difficulty). The (average) variation of a Boolean concept is defined in [15] as

$$V_n = \frac{1}{n2^n} \sum_{i=1}^{n} \sum_{neigh(x,y,i)} \delta(x,y) \ , \tag{7}$$

where the inner summation is taken over all 2^n pairs of neighboring points that differ only in their i-th attribute, and $\delta(x,y)$ equals 0 if two neighboring examples x and y belong to the same class, or equals 1 otherwise. Intuitively, V_n measures the roughness of a Boolean concept as a surface in instance space. If many pairs of neighboring examples (i.e., having Hamming distance 1 in Boolean domains) do not belong to the same class, then variation is high. Then, the connection

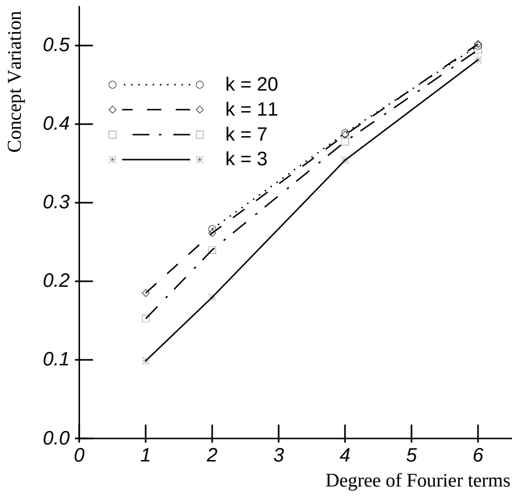

Fig. 2. Concept variation increases with Fourier complexity

between Variation and Fourier Complexity can be partly anticipated by noting that the two parity functions of n inputs have the highest variation, and the variation captured individually by each Fourier term of degree d is at most d/n (being exactly that fraction when no overlap with other Fourier terms occurs). That is, variation tends to diminish proportionally with d. Thus, functions with only a few non-zero Fourier coefficients of moderate to high degree can still be difficult to learn by most current systems due to the high variation involved in the Fourier series of such functions. Similarly, increasing the number of non-zero Fourier terms also brings up variation, but at a significantly lower rate. This is confirmed empirically as Figure 2 illustrates. All curves show a rapid increase of variation with d. Although increasing k raises each curve as a whole, this effect becomes almost imperceptible for $k > 11$.

Because the spectral norm $L(f)$ depends on the absolute value of the non-zero coefficients but not on their degrees, it is not correlated to variation. For instance, parity functions with the highest variation still have the lowest spectral norm. Thus, the spectral norm might also seem unrelated to the notion of learning difficulty as witnessed by the performance of many typical learning systems. However, results in computational learning theory strongly tie the spectral norm to learnability. In particular, any class of functions with polynomially bounded spectral norm is learnable through membership queries (under uniform distribution over the input space) [5]. Thus, it is important to find good upper bounds of spectral norm for interesting classes of functions. The difficulty of such task is acknowledged by Bellare [2], and he only gives upper bounds for simple classes (e.g., conjunctions, disjunctions, and parity trees) and classes of functions whose inductive structure simplifies the direct computation of the spectral norm (e.g. comparison functions, and each output bit in addition) . Some functions in the latter group, such as majority functions and linear thresholds on the number of bits, will be part of those used in the remaining experiments.

3 Algorithms Based on the Fourier Transform

A survey of learning methods based on the Fourier transform highlights two algorithms, one for each of two different kinds of function described next [6]. These algorithms do not rely directly on the spectral norm but on the distribution of the non-zero Fourier coefficients with respect to the degree of the terms. Thus, some Boolean functions have good approximations in terms of a few low-degree terms, and others can be accurately approximated using only a few high-degree terms. That is, the factor discriminating two relevant kinds of function is not the spectral norm itself (i.e., the sum of absolute values of Fourier coefficients) but its growth rate with respect to the coefficients' degree. For one kind of function, most of the spectral norm accumulates rapidly (on low-degree terms); for the other, the spectral norm concentrates on high-degree terms. Both cases are illustrated with the thin-line curves in Figure 3; the straight line represents a borderline between the two kinds of function and, hence, an unfavorable case for algorithms specifically targeting either kind of function. Because the spec-

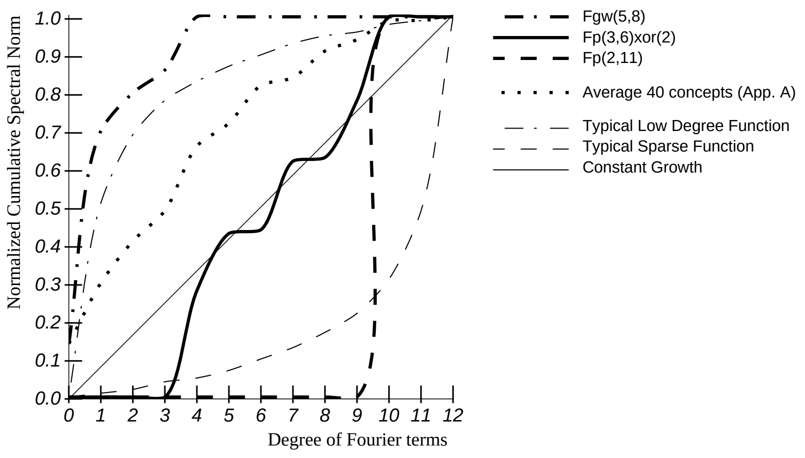

Fig. 3. Spectral norm growth for benchmark concepts from Appendix A

tral norm of different functions can vary greatly in absolute terms, we focus on the *normalized cumulative* spectral norm, using this as vertical axis in Figure 3. This axis measures the sum of *squared* Fourier coefficients (instead of their absolute values), up to a certain degree indicated by the horizontal axis. Thus, since $\sum_{z\in\{0,1\}^n}(\hat{f}(z))^2 = 1$ (as shown in [6]), the normalized cumulative norm of any function is 0 for $d = 0$, and 1 for $d = n$, regardless of how large the actual spectral norm of that function is.

For each of the two kinds of functions characterized above as extreme cases, Mansour [6] describes one learning algorithm: the Low Degree algorithm (LD), and the Sparse Function algorithm. The latter relies on membership queries, and is consequently excluded from our experiments since none of the other learners considered here uses such queries. The Low Degree algorithm uses $\mathcal{O}(n^d)$ randomly selected examples to estimate the Fourier coefficients of degree d or smaller. Recall that the Fourier coefficients can be computed by

$$\hat{f}(z) = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)\chi_z(x) \ . \tag{8}$$

However, since f is not known at learning time, the coefficients are estimated by

$$\hat{f}_{est}(z) = \frac{1}{m} \sum_{i=1}^{m} f(X_i)\chi_z(X_i) \ , \tag{9}$$

from a set of classified examples $\{X_1, X_2, \ldots, X_m\}$ randomly chosen for training. Thus, the Low Degree algorithm, LD(d), outputs the Boolean function

$$h(x) \stackrel{\text{def}}{=} sign\left(\sum_{z\in\{0,1\}^n,\ \text{degree}(z)\le d} \hat{f}_{est}(z)\chi_z(x) \ . \right) \tag{10}$$

As d increases, the number of terms to consider grows quickly, and so does the number of coefficients that need to be estimated. Consequently, increasing d means that more examples are required for accurate estimation.

Table 1. Algorithm comparison based on Fourier series controlled by k and d

Parameters of the target function's Fourier series		Predictive Accuracy					
		1% Training data			10% Training data		
		MRP	LD(d)	Difference	MRP	LD(d)	Difference
$k = 3$	$d = 2$	76.3	73.0	**3.3**	100.0	94.4	**5.6**
	$d = 4$	70.9	56.4	**14.5**	98.5	71.0	**27.5**
	$d = 6$	57.5	52.8	**4.7**	89.0	59.5	**29.5**
$k = 7$	$d = 2$	61.3	69.3	-8.0	96.1	87.9	**8.2**
	$d = 4$	52.5	55.5	-3.0	80.3	68.1	**12.2**
	$d = 6$	51.2	53.3	-2.1	68.8	58.1	**10.7**
$k = 15$	$d = 2$	56.6	67.0	-10.4	79.9	85.4	-5.5
	$d = 4$	50.2	55.2	-5.0	62.3	67.2	-4.9
	$d = 6$	49.9	52.7	-2.8	52.6	57.9	-5.3

To observe the influence of d in LD's performance, we ran LD under the same experimental design discussed in Section 2 except that now, due to LD's longer running times, only a subset of the original values of k, d, and training size are used. Here, LD is compared only to MRP, the system that was more resistant to Fourier complexity in the previous experiments. In each run, LD is provided with the actual value of d used to generate the target concept. The results are shown in Table 1, together with the corresponding results for the best performer from Section 2. It was observed in Figure 1 that the accuracy of four learners degrades quickly as d grows. Like the others, LD experiences a performance degradation for any given k. However, the accuracy degradation observed in Figure 1 for increasing k seems to affect LD to a lesser extent than the other four learning systems. As shown in Table 1, LD's resistance to increasing k is superior to MRP's, and is due to its ability to retain a large number (n^d) of Fourier terms in its hypothesis, but this also slows down its learning and classification. This feature, initially presented as beneficial, can also reduce LD's predictive accuracy when learning other concepts, such as those considered next.

4 Relational Concepts and the Fourier Transform

The above comparison of systems' performance is based on randomly generated concepts with controlled Fourier complexity. We now extend the comparison to consider the benchmark concepts used in [10] to evaluate MRP (see Appendix A). These concepts were designed to have small representation in terms of relations (some of which could be complex relations). Despite their complexity, these concepts have a clearer structure than those generated randomly for the previous experiments. Now, we characterize also these benchmark concepts in terms of their Fourier complexity. First, they all have small spectral norm, ranging from 1.0 to 17.4 and averaging 5.3 (over all 40 concepts). Thus, they should be efficiently learnable by membership queries, as suggested in [5]. However, the learning model and systems studied here do not consider exploiting the benefit of membership queries. Despite their small spectral norm, some of these concepts still have a relatively large number of Fourier terms, ranging from 1 to 512, with an overall average of 103 terms per concept.

These concepts can also be characterized with respect to how fast their spectral norm grows with respect to the degree of Fourier terms. This allows positioning each benchmark concept within the range between the two kinds of function described in Section 3 (thin lines in Figure 3). The result of such process is that these concepts are scattered throughout the entire range between the two extreme cases. To illustrate this, four curves corresponding to spectral norm growth measured on benchmark concepts are shown as thick lines in Figure 3. Three of the thick-line curves correspond to individual concepts, each one selected as a close match to one the previous curves. The fourth one corresponds to the average over all 40 benchmark concepts. This average is slightly skewed toward the case of quickly growing spectral norm (i.e., the case of functions mostly defined on low-degree terms). This suggests that on average these concepts are closer to the extreme favorable to LD than to the opposite end. However, because these concepts are scattered between both ends, LD's performance varies considerably from concept to concept.

Figure 4 summarizes how LD's accuracy varies with respect to d, for the benchmark concepts of Appendix A. All curves are based on five learning trials per concept, each one using a randomly generated training sample covering 10% of the 12-dimensional instance space. The top three graphs correspond to the same three concepts selected for Figure 3 as representatives of two extreme cases and one intermediate case of spectral norm growth. For each extreme case, LD achieves its highest accuracy at a different end of the d axis. The leftmost graph, corresponding to a concept with mostly low-degree terms, indicates a clear loss in accuracy as the complexity of the model being fit (that is, d) increases. The rightmost graph corresponds to a parity concept, that is, a concept with just one Fourier term, but a term of a high-degree. Although LD is not designed for this kind of concept, this rightmost graph serves to illustrate some problems that may affect LD in other situations as well. Note that LD's accuracy increases sharply at $d = 10$, and that the target concept is a parity function of 10 inputs. Thus, for concepts involving parity of p bits, LD(d)'s accuracy misleadingly decreases as d grows from 0 to $p - 1$. This is the same behavior observed in the leftmost graph. However, in the parity case, d must be allowed to grow more if the highest accuracy is to be reached, whereas in the leftmost graph, any increase in d was to produce only less accurate hypotheses (due to overfitting). Thus, hill-climbing approaches to grow d gradually can be misled easily.

The top middle graph in Figure 4 corresponds to a concept whose Fourier coefficients are more or less evenly distributed over mid-degree terms (from $d = 4$

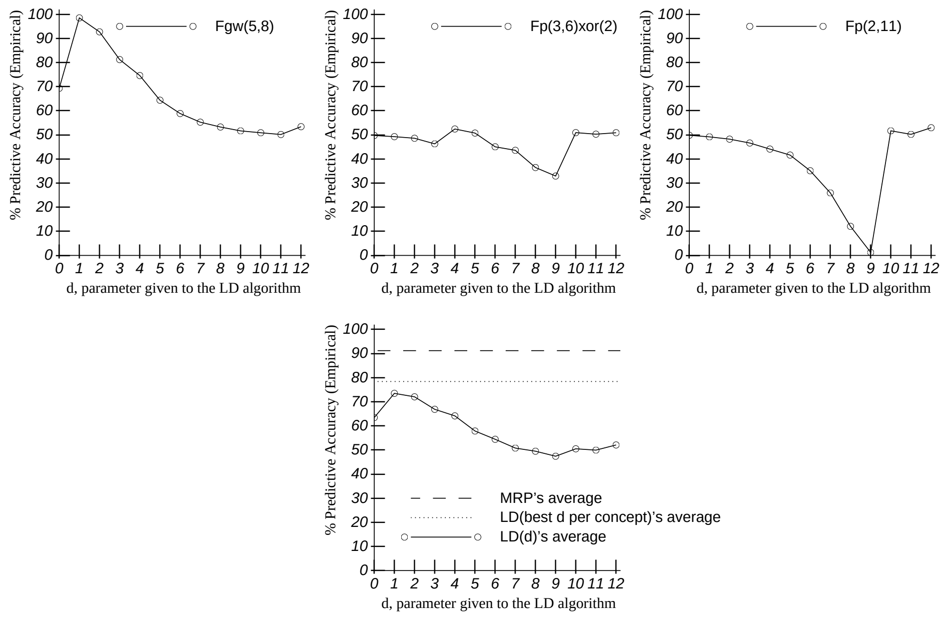

Fig. 4. LD(d)'s accuracy on benchmark concepts from Appendix A

to $d = 9$). Again, this is not the case most favorable to LD, as witnessed by its highest accuracy of about 50%. The structure of this concept is such that can be expressed concisely in terms of two relations: a parity of 4 bits and a combination of two linear threshold functions over the same set of 6 bits. Note again some sharp increases in accuracy at $d = 4$ and $d = 10(= 4 + 6)$. Thus, the previous observation about the misleading decay in accuracy as d grows also affects this concept (although to a lesser degree).

On average over these 40 concepts, LD(d)'s accuracy (bottom of Figure 4) has a smooth peak toward the low degree end, which is consistent with the observation that the average spectral norm growth is skewed toward quickly growing for small d (recall the dotted line in Figure 3). LD's average performance over these concepts is lower than MRP's, shown as dashes in the bottom graph of Figure 4. Even after selecting d^* as the best d for each benchmark concept individually, LD(d^*) still has an average accuracy (dotted line) about 10 percentage points below MRP's.

5 Discussion: Results, Extensions and Related Work

Our experiments show that LD(d) is better than other algorithms when learning low-degree functions (Figure 1 and Table 1), but worse when learning other benchmark concepts (Figure 4). In both test environments, all concepts have low spectral norm (possibly a general necessary condition for learning). However, MRP's benchmark concepts covered a wider range along distribution of spectral norm; that is, they included low-degree functions, sparse functions, and intermediate cases. As we argued in [10], these concepts were motivated by difficult real-world problems where lack of domain knowledge forces primitive representations and makes relations among attributes more relevant. Thus, for instance, molecular biologists form protein folding theories that involve complex relations among amino acids occupying consecutive positions in the protein sequence. In particular, Chou and Fasman [3] use a condition for α-helix formation that requires a sequence of 6 consecutive amino acids containing at least 4 *helix formers* (which are just amino acids known to favor the formation of helical structures in the process of protein folding). Assuming that 5 of the 20 amino acids used in proteins are helix formers, there are $\binom{6}{4} \times 5^4 \times 20^{(6-4)} = 3.75$ million sequences of amino acid satisfying the above condition. The underlying similarity among many of those sequences cannot be judged in terms of coincidences between individual amino acid positions (primitive attributes). This difficult real-world domain demands a direct analysis of relations among primitive attributes.

In other domains, such as financial markets, relevant indicators are derived from ratios and other non-linear relationships among input variables. MRP and LD take different approaches to learn in these situations. The former attempts to extract relevant relations (of any kind) from the training data available; the latter tries to construct such relations in terms of a low-degree Fourier transform. There are ways for each algorithm to mutually benefit from the other, as we will discuss next.

We have used the most straightforward implementation of the Low Degree algorithm [6]. This could be extended with parameters to control the maximum number of terms in the hypotheses, or the minimum absolute value acceptable for a Fourier coefficient to keep the corresponding term. Also, knowledge about which inputs (not just how many inputs) need to be considered in the Fourier terms can reduce running times considerably. Instead of considering all n^d terms, it would then be enough to consider the 2^d terms involving the selected attributes as suggested by [6]. Similarly, additional knowledge can constrain LD's search even further. The learner can be given knowledge of related sets of input attributes, or it can obtain such knowledge by techniques similar to those used in MRP. This type of knowledge can speedup new versions of the LD(d). On the other hand, the explicit incorporation of the Fourier basis into the search for new representations performed by algorithms like MRP can also be beneficial.

An alternative approach related to the Fourier transform and similar to the Low Degree algorithm was developed by Seshu [17]. He proposed a decision-tree learner, *R-splitter*, that could split the data using any of the 2^R parity functions (or parity features) over a subset of R variables, previously selected by the system from the original set of n input variables. Here, R is a system parameter similar to LD's d. Unlike LD, R-splitter does not express hypotheses as sums of possibly exponentially many parity features (or Fourier terms). Instead, at each decision node, R-splitter first estimates the information provided by each split based on any individual input variable, and then if such splits are not useful, it considers splitting on parity features. Although the system considers all 2^R parity features like LD does, R-splitter chooses only one parity feature to split the data at the current node, and then proceeds to recursively refine each child of the current node. Thus, R-splitter decides dynamically how many parity functions (of degree at most R) to keep in its final hypothesis. This is similar to the way MRP dynamically finds relations in the training data, but MRP is not limited to using parity relations. It extracts the extensional representation of the relevant relations from the data.

The second basic algorithm reviewed in [6] is the Sparse Function algorithm [5]. It learns functions that can be accurately approximated by a small number of Fourier coefficients corresponding to high-degree terms. Both algorithms, Low Degree and Sparse Function, learn classes of Boolean functions *in terms of* a different class [12], namely, the class of real functions. The algorithms are not restricted to output a Boolean function, that is, one whose only possible values are -1 and $+1$. Instead, they output a real function to be interpreted as Boolean (see Section 1). This suggests that separating the two Boolean values at zero may not always be the best choice [7]. A new question related to this issue is whether the method used to estimate the Fourier coefficient is always the best. Both algorithms base their coefficient estimation in the (canonical) definition of the coefficients given by

$$\hat{f}(z) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)\chi_z(x) \ . \tag{11}$$

This definition forces the Fourier transform to be a strictly Boolean function. Thus, for instance, the function *majority of four inputs* (which returns true if and only at least two of its inputs are 1) has the following Fourier series:

$$\begin{aligned} maj_4(x_1,x_2,x_3,x_4) = \\ &+0.375\chi_{0000}(x) - 0.375\chi_{0001}(x) - 0.375\chi_{0010}(x) - 0.125\chi_{0011}(x) \\ &-0.375\chi_{0100}(x) - 0.125\chi_{0101}(x) - 0.125\chi_{0110}(x) + 0.125\chi_{0111}(x) \\ &-0.375\chi_{1000}(x) - 0.125\chi_{1001}(x) - 0.125\chi_{1010}(x) + 0.125\chi_{1011}(x) \\ &-0.125\chi_{1100}(x) + 0.125\chi_{1101}(x) + 0.125\chi_{1110}(x) + 0.375\chi_{1111}(x) \ . \end{aligned}$$

However, being a linear threshold, this function can be expressed more concisely in terms of low-degree coefficients only, as follows:

$$maj_4(x_1,x_2,x_3,x_4) = -0.5\chi_{0001}(x) - 0.5\chi_{0010}(x) - 0.5\chi_{0100}(x) - 0.5\chi_{1000}(x) \ .$$

These series are two different real functions, but the same Boolean function (when considering only their sign). Learners based on estimation of Fourier coefficients should be prepared to choose among competing sets of coefficients.

6 Conclusion

This paper has brought together approaches from the machine learning community and the computational learning theory community. The unifying context was to explore the use of new representations that facilitate learning from primitive attributes, when complex relationships are involved. The Fourier transform suggests new measures of learning difficult, and provides a way to express structured, albeit complex, concepts as simple combinations of the parity basis functions. However, without the help of membership queries, current algorithms for learning in terms of the Fourier transform perform better than other learning algorithms only for Low Degree functions. Despite the small spectral norm of the benchmark concepts from [10], LD(d) did not learn then more accurately than MRP, on average. LD(d) was not able to extract the relevant relation because it insisted on expressing them in terms of low-degree Fourier terms. The simple structure of these concepts, when expressed in terms of the appropiate relations, make them realistic in the context of difficult real-world domains where only primitive input representations are available. Our discussion of empirical results relates concept characteristics and learning algorithm design, suggesting directions to improve current learning systems.

Acknowledgments

This research was supported in part by grant IRI 92-04473 from the US National Science Foundation. We thank Lenny Pitt for bringing to our attention the literature on learning through the Fourier transform, and for his suggestions.

References

1. Yaser S. Abu-Mostafa. Complexity of random problems. In Yaser S. Abu-Mostafa, editor, *Complexity in Information Theory*, chapter VI, pages 115–131. Springer-Verlag, NY, 1988.
2. Mihir Bellare. A technique for upper bounding the spectral norm with applications to learning. In *Proc. of the Fifth Annual Conf. on Computational Learning Theory*, pages 62–70, 1992.
3. Peter Y. Chou and Gerald D. Fasman. Prediction of protein conformation. *Biochemistry*, 13(2):222–245, 1974.
4. Li-Min Fu and Bruce G. Buchanan. Learning intermediate concepts in constructing a hierarchical knowledge base. In *Proc. of the Ninth Int. Joint Conf. on Artificial Intelligence*, pages 659–666, 1985.
5. Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier transform. In *Proc. of the* 23^{rd} *Annual ACM Symp. on Theory of Computing*, pages 455–464. ACM Press, 1991.
6. Yishay Mansour. Learning boolean functions *via* the fourier transform. In V. P. Roychodhury, K-Y. Siu, and A. Orlitsky, editors, *Advances in Neural Computation*, pages 151–155. Kluwer Academic Publishers, 1994.
7. Yishay Mansour and Sahar Sigal. Implementation isssues in the fourier transform algorithm. In *Proc. of the Neural Information Processing Systems, Natural and Synthetic*, 1995.
8. Tom M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Computer Science Department, Rutgers University, New Brunswick, NJ 08903, May 1980.

9. Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.
10. Eduardo Pérez and Larry A. Rendell. Using multidimensional projection to find relations. In *Proc. of the 12th Int. Conf. on Machine Learning*, pages 447–455. Morgan Kaufmann Publishers, Inc., 1995.
11. Eduardo Pérez and Larry A. Rendell. Learning despite concept variation by finding structure in attribute-based data. In *Proc. of the 13th Int. Conf. on Machine Learning*, pages 391–399. Morgan Kaufmann Publishers, Inc., 1996.
12. Leonard Pitt and Leslie G. Valiant. Computational limitations of learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984, October 1988.
13. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., Palo Alto, CA, 1993.
14. Larry Rendell. A general framework for induction and a study of selective induction. *Machine Learning*, 1(2):177–226, 1986.
15. Larry A. Rendell and Raj Seshu. Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence*, 6:247–270, 1990.
16. Cullen Schaffer. A conservation law for generalization performance. In *Proc. of the Eleventh Int. Conf. on Machine Learning*, pages 259–265, 1994.
17. Raj Seshu. Solving the parity problem. In *Proc. of the 4th European Working Session on Learning*, pages 263–271, Montpellier, France, December 1989.
18. Satosi Watanabe. *Knowing and Guessing.* John Wiley & Sons, New York, 1969.

Appendix A. Benchmark Concepts

The following concepts, used as benchmark in [10], are members of the groups described below, where the notation $x_{i..j}$ is shorthand for $x_i, x_{i+1}, x_{i+2}, ..., x_j$:

$F_{p(2,11)}$	$F_{p(3,10)}$	$F_{p(4,9)}$	$F_{p(5,8)}$	$F_{cp(2,11)}$	$F_{cp(3,10)}$
$F_{cp(4,9)}$	$F_{cp(5,8)}$	$F_{cdp(1,9)}$	$F_{cdp(2,10)}$	$F_{cdp(3,11)}$	$F_{p(3,6)\wedge(2)}$
$F_{p(3,6)\vee(2)}$	$F_{p(3,6)\oplus(2)}$	$F_{p(3,6)\wedge(3)}$	$F_{p(3,6)\vee(3)}$	$F_{p(3,6)\oplus(3)}$	$F_{p(3,6)\wedge(2\text{–or–}3)}$
$F_{p(3,6)\vee(2\text{–or–}3)}$	$F_{p(3,6)\oplus(2\text{–or–}3)}$	$F_{mj(4,8)}$	$F_{mj(3,9)}$	$F_{mj(2,10)}$	F_{mx6}
$F_{mx6c(6,7)}$	$F_{mx6c(5,8)}$	$F_{mx6c(4,9)}$	$F_{mx6c(3,10)}$	$F_{rk(5,7)}$	$F_{rk(6,7)}$
$F_{rk(6,9)}$	$F_{rk(7,9)}$	$F_{nm(4,5,7)}$	$F_{nm(5,6,9)}$	$F_{gw(3,10)}$	$F_{gw(4,9)}$
$F_{gw(5,8)}$	$F_{sw(3,10)}$	$F_{sw(4,9)}$	$F_{sw(5,8)}$		

Parity. Let odd(s)=true *iff* an odd number of bits in s are 1. Then,
$F_{p(i,j)} \overset{\text{def}}{=} \text{odd}(x_{i..j})$, $F_{cp(i,j)} \overset{\text{def}}{=} \text{odd}(x_{i..6}) \wedge \text{odd}(x_{7..j})$, and
$F_{cdp(i,j)} \overset{\text{def}}{=} \text{odd}(x_{i..4}) \wedge [\text{odd}(x_{i+j/2..8}) \vee \text{odd}(x_{j..12})]$
Majority. $F_{mj(i,j)} \overset{\text{def}}{=} \text{maj}(x_{i..j})$, maj($s$)=true *iff* at least half of the bits in s are 1.
Parity and counters. Let l-in(s)=true *iff* exactly l bits in s are 1. Then, for $l \in \{2, 3, \text{2-or-3}\}$ and $\nabla \in \{\wedge, \vee, \oplus\}$, $F_{p(i,j)\nabla(l)} \overset{\text{def}}{=} \text{odd}(x_{i..j}) \, \nabla \, l\text{-in}(x_{7..12})$.
Multiplexors. Let $\text{mx}(i, j, d_0, d_1, d_2, d_3) = d_{2i+j}$, $\text{mx}_c(0, 0, d_{0..i}) = \bigwedge(d_{0..i})$,
$\text{mx}_c(0, 1, d_{0..i}) = \bigvee(d_{0..i})$, $\text{mx}_c(1, 0, d_{0..i}) = \bigoplus(d_{0..i})$, and $\text{mx}_c(1, 1, d_{0..i}) = \neg \bigoplus(d_{0..i})$.
Then, $F_{mx6} \overset{\text{def}}{=} \text{mx}(x_1, x_2, x_3, x_6, x_9, x_{12})$, and $F_{mx6c(i,j)} \overset{\text{def}}{=} \text{mx}_c(x_1, x_2, x_{i..j})$
Linear thresholds. Let w(s) be the number of ones in s. Then, we define $F_{rk(i,k)} \overset{\text{def}}{=} \text{w}(x_{6-\lfloor k/2 \rfloor..7+\lfloor k/2 \rfloor}) = i$, $F_{nm(i,j,k)} \overset{\text{def}}{=} \text{w}(x_{6-\lfloor k/2 \rfloor..7+\lfloor k/2 \rfloor}) \in \{i, j\}$,
$F_{gw(i,j)} \overset{\text{def}}{=} \text{w}(x_{i..6}) > \text{w}(x_{7..j})$, and $F_{sw(i,j)} \overset{\text{def}}{=} \text{w}(x_{i..6}) = \text{w}(x_{7..j})$.