On the exact learnability of graph parameters*

Nadia Labai †

Department of Informatics, TU Wien, Vienna, Austria
labai@dbai.tuwien.ac.at

In this talk, we discuss a new direction for research in learning theory and present proof-of-concept.

In the exact learning model, [1], the learner wishes to compute an exact representation of some target function f in some class $\mathcal{C}$. For this purpose, the learner may query an all-powerful teacher with two kinds of queries:

1. VALUE(x) queries, where the learner sends input x and the teacher returns $f(x)$, and
2. EQUIVALENT(h) queries, where the learner proposes a hypothesis h and if the hypothesis is correct, the teacher returns "YES" and if it is incorrect, the teacher returns a counterexample. That is, some input x such that $h(x) \neq f(x)$.

The class $\mathcal{C}$ is *exactly learnable* if there is a learner that computes a correct hypothesis in time polynomial in the size of the smallest representation of f and the size of the largest counterexample returned by the teacher.

Exact learning and Hankel matrices The exact learnability of word and tree functions representable by weighted automata has been established for various domains, [3, 10, 2].

Exact learning algorithms for weighted automata usually rely on an algebraic characterization of the functions they compute, using Hankel matrices. Let Σ be a finite alphabet and let $f : \Sigma^\star \to \mathcal{R}$ be a word function, where $\mathcal{R}$ is a field or a semiring. The *Hankel matrix* $\mathbf{H}_f$ is a bi-infinite matrix where the rows and the columns are labeled with words in $\Sigma^\star$ and where the entry at the row labeled u and column labeled v holds $f(uv)$. That is, $\mathbf{H}_f \in \mathcal{R}^{\Sigma^\star \times \Sigma^\star}$ and $\mathbf{H}_f(u, v) = f(uv)$. The definition for tree functions is similar; the rows are labeled with trees and the columns are labeled with contexts.

A typical algebraic characterization theorem relates the rank of $\mathbf{H}_f$ to whether f can be represented by a weighted automaton, and its proof usually provides a translation from $\mathbf{H}_f$ to the automaton which computes f, [4, 7, 5]. Then the learning algorithm may iteratively build a submatrix of $\mathbf{H}_f$ using query answers until eventually the submatrix grows to have rank large enough to extract an automaton that computes f.

*This is an extended abstract of [13], which was done in collaboration with Johann A. Makowsky (Department of Computer Science, Technion IIT, Israel).

†Supported by the LogiCS doctoral program (W1255) funded by the Austrian Science Fund (FWF).

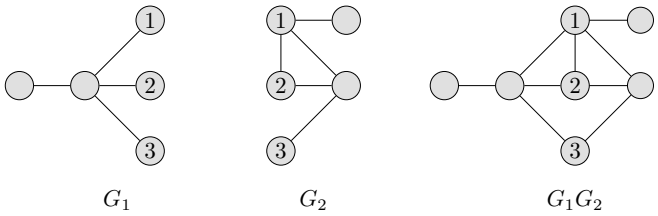

Figure 1: Two 3-labeled graphs G_1 and G_2, and their 3-connection G_1G_2

Exactly learning graph parameters A *graph parameter* is a function that assigns graphs values in $\mathcal{R}$ which is isomorphism invariant. In our context, $\mathcal{R}$ is $\mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$. Hankel matrices can be defined for graph parameters using various graph operations instead of concatenation. In particular, connection matrices are defined using the connection operation. A *k-labeled graph* G for $k \in \mathbb{N}$ is a finite graph in which k vertices, or less, are labeled with labels from $\{1, \ldots, k\}$. The *k-connection* of two k-labeled graphs G_1, G_2 is given by taking the disjoint union of G_1 and G_2 and identifying vertices with the same label. This produces a k-labeled graph $G = G_1G_2$. An example is shown in figure 1.

Denote the set of graphs by $\mathcal{G}$ and the set of k-labeled graphs by $\mathcal{G}_k$. Given a graph parameter $f : \mathcal{G} \to \mathcal{R}$, the *$k$-connection matrix* $\mathbf{C}_{f,k}$ is a bi-inifinite matrix over $\mathcal{R}$ whose rows and columns are labeled with k-labeled graphs, where the entry at the row labeled G_i and the column labeled G_j holds $f(G_iG_j)$. That is, $\mathbf{C}_{f,k} \in \mathcal{R}^{\mathcal{G}_k \times \mathcal{G}_k}$ and $\mathbf{C}_{f,k}(G_i, G_j) = f(G_iG_j)$.

In [9], the rank of $\mathbf{C}_{f,k}$ has been related to whether f is MSOL-definable. For the notion of MSOL-defianability for graph parameters, see [16]. This encourages us to investigate the exact learnability of MSOL-definable graph parameters. However, as opposed to the algebraic characterization theorems for weighted automata, the result in [9] is not a characterization, but more importantly, its proof does not provide a translation from $\mathbf{C}_{f,k}$ to the MSOL-expression defining f. We therefore restrict our attention to the smaller class of graph parameters representable as a partition function.

The case of partition functions Partition functions, aka counting weighted homomorphism functions, are a subclass of the MSOL-definable graph parameters.

A weighted graph $H(\alpha, \beta)$ is a graph $H = (V(H), E(H))$ on $n = |V(H)|$ vertices together with a vertex weight function $\alpha : V(H) \to \mathbb{R}$, viewed as a vector of length n, and an edge weights function $\beta : V(H)^2 \to \mathbb{R}$ viewed as an $n \times n$ matrix, with $\beta(u, v) = 0$ if $(u, v) \notin E(H)$.

$H(\alpha, \beta)$ defines a partition function $\mathsf{hom}(-, H(\alpha, \beta))$, whose value on a graph G is defined as follows:

$$\mathsf{hom}(G, H(\alpha, \beta)) = \sum_{t:G \to H} \prod_{v \in V(G)} \alpha(t(v)) \prod_{(u,v) \in V(G)^2} \beta(t(u), t(v))$$

For example, the graph $H_{\text{edge-cover}}$ depicted in figure 2 defines a function which counts the number of covering edge sets, and $H_{3\text{-col}}$, depicted in figure 3, defines a function which counts the number of 3-colorings.

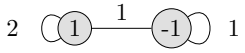

Figure 2: $H_{\text{edge-cover}}$

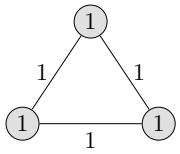

Figure 3: $H_{\text{3-col}}$

A weighted graph $H(\alpha, \beta)$ is said to be *twin-free* if β does not contain two separate rows that are identical to each other. We say a partition function $\mathsf{hom}(-, H(\alpha, \beta))$ is *rigid* if H has no proper automorphisms. Note that automorphisms in a weighted graph also respect vertex and edge weights. In our examples above, $H_{\text{edge-cover}}$ is rigid and $H_{\text{3-col}}$ is not.

An extensive body of work on the graph algebras induced by connection operations and partition functions, [8, 14, 15], sets up the following result:

Theorem 1 (Lovász, [14])**.** *Let $f = \mathsf{hom}(-, H(\alpha, \beta))$ for a rigid twin-free weighted graph $H(\alpha, \beta)$ on q vertices. Then $\mathbf{C}_{f,k}$ has rank q^k for all $k \geq 0$.*

It still remains to find a translation from $\mathbf{C}_{f,k}$ to the corresponding weighted graph $H(\alpha, \beta)$. This is done by identifying the suitable algebraic properties underlying the proof of Theorem 1 and extracting them through somewhat technical algebraic manipulations.

With the translation procedure in hand, we design a typical learning algorithm: it maintains a submatrix M of $\mathbf{C}_{f,1}$ used in the generation of the hypothesis h from VALUE and EQUIVALENT query results. After an initial setup of M, in each iteration the algorithm generates a hypothesis h, queries the teacher for equivalence between h and the target and either terminates, or augments M accordingly and moves on to the next iteration. It is guaranteed that the rank of M increases in each iteration, which implies the algorithm terminates successfully after q iterations. Since each iteration takes time polynomial in the size of the target weighted graph and the largest counterexample, we have:

Theorem 2 (L and Makowsky, [13])**.** *The class of rigid partition functions is exactly learnable.*

We assume to be over the Blum-Shub-Smale model of computation. If f takes values in $\mathbb{Q}$ rather than in $\mathbb{R}$ we can also work in the Turing model with logarithmic cost for the elements in $\mathbb{Q}$.

Limitation to rigid graphs The target weighted graph is assumed to be rigid. We note that almost all graphs are rigid:

Theorem 3 ([6, 11])**.** *Let G be a uniformly selected graph on n vertices. The probability that G is rigid tends to 1 as $n \to \infty$.*

Lifting the rigidity restriction would require the algorithm to keep a submatrix of the k-connection matrix, where k is determined by the target weighted graph and is therefore unknown. However, if we can find the correct k quickly enough, the same algorithm should apply.

How powerful does the teacher need to be? Computing the value of $\mathsf{hom}(G, H(\alpha, \beta))$ is generally intractable (contains #P-hard problems, such as counting colorings), but is in function polynomial time (FP) if G is of bounded tree-width, [15, Theorem 6.48], or of bounded clique-width, [12].

As for answering EQUIVALENT queries, from [15, Theorem 6.45], we have that the counterexamples provided by the teacher may be chosen to be of size at most $2(1 + q^2)q^6$ where q is the size of the target weighted graph. Testing whether a hypothesis is correct reduces to the weighted graph isomorphism problem, which is at least as difficult as the unweighted version. It is unknown whether this problem is in PTIME.

References

[1] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39(3):337–350, 1978.

[2] B. Balle and M. Mohri. *Learning Weighted Automata*, pages 1–21. Springer International Publishing, 2015.

[3] A. Beimel, F. Bergadano, N.H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *Journal of the ACM (JACM)*, 47(3):506–530, 2000.

[4] S. Bozapalidis and O. Louscou-Bozapalidou. The rank of a formal tree power series. *Theoretical Computer Science*, 27(1):211 – 215, 1983.

[5] J.W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5:26–40, 1971.

[6] P. Erdős and A. Rényi. Asymmetric graphs. *Acta Mathematica Hungarica*, 14(3-4):295–315, 1963.

[7] M. Fliess. Matrices de hankel. *Journal de Mathématiques Pures et Appliquées*, 53(9):197–222, 1974.

[8] M. Freedman, L. Lovász, and A. Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *Journal of the American Mathematical Society*, 20(1):37–51, 2007.

[9] B. Godlin, T. Kotek, and J.A. Makowsky. Evaluation of graph polynomials. In *34th International Workshop on Graph-Theoretic Concepts in Computer Science, WG08*, volume 5344 of *Lecture Notes in Computer Science*, pages 183–194, 2008.

[10] A. Habrard and J. Oncina. Learning multiplicity tree automata. In *Grammatical Inference: Algorithms and Applications*, pages 268–280. Springer, 2006.

[11] J. Kötters. Almost all graphs are rigid - revisited. *Discrete Mathematics*, 309(17):5420–5424, 2009.

[12] N. Labai and J.A. Makowsky. Tropical Graph Parameters. In *26th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2014)*, DMTCS Proceedings, pages 357–368. Discrete Mathematics and Theoretical Computer Science, 2014.

[13] N. Labai and J.A. Makowsky. On the exact learnability of graph parameters: The case of partition functions. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016*, pages 63:1–63:13, 2016.

[14] L. Lovász. The rank of connection matrices and the dimension of graph algebras. *European Journal of Combinatorics*, 27(6):962 – 970, 2006.

[15] L. Lovász. *Large Networks and Graph Limits*, volume 60 of *Colloquium Publications*. AMS, 2012.

[16] J.A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126.1-3:159–213, 2004.