

Exact Learning Multiplicity Automata

Nader H. Bshouty

Eyal Kushilevitz

Abstract

Here we will give an algorithm for learning Multiplicity Automata.

1 Learning using the Hankel Matrix Representation

In this section we show how to learn efficiently more functions using membership queries and equivalence queries. This time we use the Hankel matrix representation introduced in Section ???. Note that the size of such a matrix F , corresponding to a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, is exponential in n independent of the “complexity” of the function f that it represents (since F has a row and a column corresponding to *every* string of length at most n). As indicated in Section ??, in the context of the Hankel matrix representation, we measure the complexity of a function f by the *rank* of the corresponding Hankel matrix F .

The idea is as follows. Instead of explicitly creating a copy of F we will have at any given step an *implicit* representation of F . This implicit representation includes a matrix H which is an $r \times r$ sub-matrix of F , the Hankel matrix corresponding to f , and some information about the linear dependencies that should allow computing the value of f for inputs which are not in H . The rows of H are indexed by strings $X = \{x_1, \dots, x_r\}$ and the columns of H are indexed by strings $Y = \{y_1, \dots, y_r\}$ (note that based on X and Y the matrix H can be generated using r^2 membership queries). We will maintain the property that H is of full rank (i.e., it has rank r). Obviously, the rank of H is bounded by the rank of F . The algorithm will start by asking EQ(0). If the target function is identically 0 we are done. Otherwise, we will get a counterexample z such that $f(z) = 1$. We initialize $x_1 = \epsilon, x_2 = z, y_1 = \epsilon$ and $y_2 = z$ so we get a 2×2 matrix H of rank 2. That is,

$$H = \begin{array}{c|cc} & \epsilon & z \\ \hline \epsilon & 0 & 1 \\ z & 1 & 0 \end{array}$$

The structure of the algorithm is as follows. Given the current H , we show below how to construct a hypothesis h such that from a counterexample z satisfying $f(z) \neq h(z)$ we can always find a row and a column such that adding them to H increases the rank by 1. If we can do this then after at most $\text{rank}(F)$ iterations we are done.

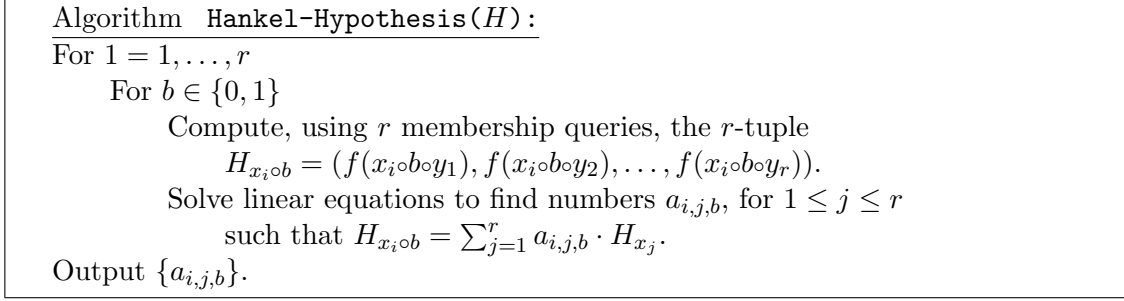


Figure 1: Algorithm Hankel-Hypothesis(H)

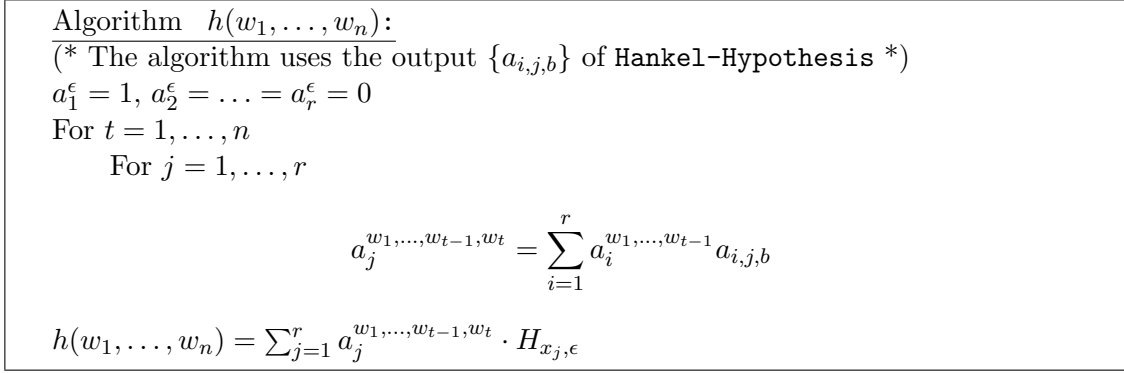


Figure 2: Algorithm $h(w_1, \dots, w_n)$

In Figure ?? we show an algorithm, called **Hankel-Hypothesis** that generates a hypothesis h based on H . It does so by asking some additional membership queries (but no equivalence query) and outputs a set of numbers $\{a_{i,j,b}\}$. In Figure ?? we show how the value $h(w_1, \dots, w_n)$ is actually computed based on these numbers. We now explain how these algorithms work. We look, for every $x_i \in X$ and $b \in \{0, 1\}$, at the r -tuple corresponding to the entries of F in row $x_i \circ b$ and columns in Y . Denote this r -tuple by $H_{x_i \circ b}$ (the tuple $H_{x_i \circ b}$ is generated using r membership queries). That is,

$$H_{x_i \circ b} = (f(x_i \circ b \circ y_1), f(x_i \circ b \circ y_2), \dots, f(x_i \circ b \circ y_r)).$$

Since H is a full rank matrix we can find (efficiently) coefficients $a_{i,j,b}$ such that

$$H_{x_i \circ b} = \sum_{j=1}^r a_{i,j,b} \cdot H_{x_j}.$$

Now, given w , we can compute $h(w)$ as follows. We use the numbers $\{a_{i,j,b}\}$ which were computed as the coefficients of the above $2r$ linear equations to express (efficiently) every

vector H_w as a linear combination of H_{x_1}, \dots, H_{x_r} (this representation of H_w is *not* necessarily correct). This is done by induction; $w = \epsilon$ is easy since $x_1 = \epsilon$ (and so $H_\epsilon = H_{x_1}$). Now, if we already expressed $H_w = \sum_{i=1}^r a_i^w H_{x_i}$ then we write $H_{w \circ b} = \sum_{i=1}^r a_i^w H_{x_i \circ b}$. Notice that if a_i^w are the “correct” coefficients, i.e., $F_w = \sum_{i=1}^r a_i^w F_{x_i}$, then $F_{w \circ b} = \sum_{i=1}^r a_i^w F_{x_i \circ b}$ (since by the definition of Hankel matrices $F_{w \circ b, v} = F_{w, b \circ v}$). Using the coefficients $\{a_{i,j,b}\}$ we get

$$H_{w \circ b} = \sum_{i=1}^r a_i^w \left[\sum_{j=1}^r a_{i,j,b} H_{x_j} \right] = \sum_{j=1}^r \left[\left(\sum_{i=1}^r a_i^w a_{i,j,b} \right) \cdot H_{x_j} \right],$$

so the coefficient $a_j^{w \circ b}$ is computed by $\sum_{i=1}^r a_i^w \cdot a_{i,j,b}$. Finally, for every w , since we can compute H_w then in particular we can efficiently compute $h(w) \triangleq H_{w,\epsilon}$ (note that if H_w is “correct”, i.e., H_w indeed contains the correct values as in F , then in particular $H_{w,\epsilon} = F_{w,\epsilon} = f(w)$).

Algorithm Learn-Hankel:

(* Initialization *)

Ask EQ(0)

If YES output 0. Otherwise, z is a counterexample.

Set $x_1 = \epsilon, x_2 = z, y_1 = \epsilon$ and $y_2 = z$ and H the corresponding 2×2 matrix.

(* Main Loop *)

While(TRUE)

$h \leftarrow \text{Hankel-Hypothesis}(H)$

 Ask EQ(h). If YES output h . Otherwise z is a counterexample

 Find the first k such that $H_{z_1, \dots, z_k, z_{k+1}} \neq \sum_{j=1}^k a_j^{z_1, \dots, z_k} H_{x_j \circ z_{k+1}}$ (where the $a_j^{z_1, \dots, z_k}$ are computed as in Figure ??)

 Find $y \in Y$ such that $H_{z_1, \dots, z_k}(z_{k+1} \circ y) \neq \sum_{j=1}^k a_j^{z_1, \dots, z_k} H_{x_j}(z_{k+1} \circ y)$

$x_{r+1} \leftarrow z_1, \dots, z_k, y_{r+1} \leftarrow z_{k+1} \circ y$

Figure 3: Algorithm Learn-Hankel

After constructing the hypothesis h we ask an equivalence query EQ(h). Either we get the answer YES or a counterexample z such that $f(z) \neq h(z)$. We claim that z can be decomposed to $z = w \circ b \circ v$ such that adding a row w to X and a column $b \circ y$ (for some $y \in Y$ that we will find) to Y will increase the rank of H by 1. This decomposition can be found by considering all prefixes w of z . To see this, observe that there must exist a prefix w for which the coefficients a_j^w computed by the above algorithm are correct, i.e., $F_w = \sum_{j=1}^r a_j^w F_{x_j}$ (this is tested by the algorithm by verifying that $H_w = \sum_{j=1}^r a_j^w H_{x_j}$), but for these coefficients $H_{w \circ b} \neq \sum_{i=1}^r a_i^w H_{x_i \circ b}$. (For $w = \epsilon$ the coefficients are certainly correct and if no such w exists it follows, by induction, that the coefficients of H_z are also correct which implies that $h(z) = f(z)$ – a contradiction.) In particular, for such w , there exists $y \in Y$ such that $H_w(b \circ y) \neq \sum_{i=1}^r a_i^w H_{x_i}(b \circ y)$. It follows that if we add the column $b \circ y$ to Y then the row H_w must be independent of the previous rows H_{x_1}, \dots, H_{x_r} , as needed (since if H_w depends on H_{x_1}, \dots, H_{x_r} the coefficients must be a_1^w, \dots, a_r^w but adding $b \circ y$ to Y eliminates this possibility).

Exercise 1.0.1 *Analyze the complexity of the procedure `Hankel-Hypothesis`. (In particular, how many membership queries it uses).*

Analyze the complexity of algorithm `Learn-Hankel`

Exercise 1.0.2 *Improve the complexity of algorithm `Learn-Hankel` by finding the prefix w of z using an appropriate binary search.*

Exercise 1.0.3 *General alphabets.*

Exercise 1.0.4 *deterministic automata*