Exactly Learning Weighted Automata over a Field

Preliminaries

A weighted automaton over a field $\mathbb{K}$ is a tuple $\mathcal{A} = (n, \Sigma, \boldsymbol{\alpha}, \{M(\sigma)\}_{\sigma\in\Sigma}, \boldsymbol{\eta})$ comprising the *dimension* $n \in \mathbb{N}$, *alphabet* Σ, *initial-state vector* $\boldsymbol{\alpha} \in \mathbb{K}^n$, family of *transition matrices* $M(\sigma) \in \mathbb{K}^{n\times n}$, and *final-state vector* $\boldsymbol{\eta} \in \mathbb{K}^n$. Extend M freely to Σ^* by writing $M(\sigma_1 \ldots \sigma_k) = M(\sigma_1)\cdots M(\sigma_k)$. Then $\mathcal{A}$ is said to *recognize* a formal power series $f : \Sigma^* \to \mathbb{K}$ if $f(w) = \boldsymbol{\alpha}^T M(w)\boldsymbol{\eta}$ for all $w \in \Sigma^*$.

Write $e_i \in \mathbb{K}^n$ for the column vector with 1 in the i-th position and 0 in all other positions.

Define the *Hankel matrix* of a formal power series $f : \Sigma^* \to \mathbb{K}$ to be the infinite matrix F whose rows and columns are indexed by Σ^*, such that $F_{x,y} = f(xy)$ for $x, y \in \Sigma^*$. Recall that if f is recognized by a $\mathbb{K}$-weighted automaton $\mathcal{A}$ then the rank of its Hankel matrix is at most the number of states of $\mathcal{A}$.

The Algorithm

We describe an algorithm (from [1]) to exactly learn a weighted automaton computing a given function $f : \Sigma^* \to \mathbb{K}$ using membership and equivalence queries. In a membership query the learner asks for the value of f on a given word $w \in \Sigma^*$.

At each stage the algorithm maintains the following data:

- A set of n "rows" $X = \{x_1, \ldots, x_n\} \subseteq \Sigma^*$, where $x_1 = \varepsilon$.
- A set of n "columns" $Y = \{y_1, \ldots, y_n\} \subseteq \Sigma^*$, where $y_1 = \varepsilon$.
- A full-rank $n \times n$ submatrix H of F, determined by X and Y:

$$H = \begin{bmatrix} f(x_1y_1) & f(x_1y_2) & \cdots & f(x_1y_n) \\ f(x_2y_1) & f(x_2y_2) & \cdots & f(x_2y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_ny_1) & f(x_ny_2) & \cdots & f(x_ny_n) \end{bmatrix}$$

The entries of the matrix H are determined by making membership queries.

These data determine a *Hypothesis automaton* $\mathcal{A}$ as follows. Intuitively the states of $\mathcal{A}$ correspond to the rows of H, with the i-th row being the state reached after executing word x_i from the initial state. The columns can be considered as tests that distinguish different states.

Formally $\mathcal{A}$ has dimension n, initial-state vector $\alpha = e_1^T H$, the first row of H, and final-state vector $\eta = e_1$. Since H has full rank, for each $\sigma \in \Sigma$ we can define the transition matrix $M(\sigma)$ by the equation

$$HM(\sigma) = \begin{bmatrix} f(x_1\sigma y_1) & f(x_1\sigma y_2) & \cdots & f(x_1\sigma y_n) \\ f(x_2\sigma y_1) & f(x_2\sigma y_2) & \cdots & f(x_2\sigma y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_n\sigma y_1) & f(x_n\sigma y_2) & \cdots & f(x_n\sigma y_n) \end{bmatrix}$$

In each step of the algorithm an equivalence query is performed to determine whether $\mathcal{A}$ computes f. If not, a counterexample $w \in \Sigma^*$ is returned.

Proposition 1 *A counterexample z has a prefix $w\sigma$, where $\sigma \in \Sigma$ and $w \in \Sigma^*$, such that for some $i \in \{1, \ldots, n\}$ the assignment $X \leftarrow X \cup \{w\}$, $Y \leftarrow Y \cup \{\sigma y_i\}$ increases the rank of H by one.*

Proof. Say that automaton $\mathcal{A}$ is *correct* on a word $w \in \Sigma^*$ if

$$\alpha M(w) = (f(wy_1), \ldots, f(wy_n)) \,. \tag{1}$$

Note that in this case $\mathcal{A}(w) = \alpha M(w)\eta = f(w)$. It follows that $\mathcal{A}$ is not correct on z. Since it is clearly correct on the empty word, there must exist a prefix $w\sigma$ of z such that $\mathcal{A}$ is correct on w, but not on $w\sigma$. For such a w we have that (1) holds, but also

$$\alpha M(w\sigma) \neq (f(w\sigma y_1), \ldots f(w\sigma y_n)) \,.$$

In particular, we can pick $i \in \{1, \ldots, n\}$ such that

$$\alpha M(w\sigma) e_i \neq f(w\sigma y_i) \,. \tag{2}$$

Now consider the matrix H' defined by

$$H' = \begin{bmatrix} f(x_1 y_1) & f(x_1 y_2) & \cdots & f(x_1 y_n) & f(x_1\sigma y_i) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f(x_n y_1) & f(x_n y_2) & \cdots & f(x_n y_n) & f(x_n\sigma y_i) \\ f(wy_1) & f(wy_2) & \ldots & f(wy_n) & f(w\sigma y_i) \end{bmatrix}$$
$$\stackrel{(1)}{=} \begin{bmatrix} H & HM(\sigma)e_i \\ \alpha M(w) & f(w\sigma y_i) \end{bmatrix} .$$

It remains to show that H' has rank $n+1$. By assumption H has rank n, so it suffices to show that the $(n+1)$-st row of H' cannot be expressed as a linear combination of the first n rows. Indeed, suppose for a contradiction that $u \in \mathbb{K}^n$ is such that $u^T H = \alpha M(w)$ and $u^T HM(\sigma)e_i = f(w\sigma y_i)$. Then

$$f(w\sigma y_i) = u^T HM(\sigma)e_i = \alpha M(w)M(\sigma)e_i\,,$$

which contradicts (2). $\square$

The word w and suffix σy_i in the above proposition can be found using membership queries.

Comparison with Angluin's Algorithm

The rows and columns in the above algorithm play a similar role to the access words and test words in Angluin's algorithm. The requirement that H have full rank corresponds to the conditions of closedness and separatedness in Angluin's algorithm. Intuitively the situation for weighted automata is more symmetric than for DFA: in particular, the number of rows and columns is always the same.

References

[1] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47:2000, 2000.