UN ALGORITMO DE INFERENCIA DE LENGUAJES REGULARES USANDO DATOS POSITIVOS Y NEGATIVOS

J. Oncina, P. García
Dpto. de Sistemas Informáticos y Computación. U.P. de Valencia

RESUMEN

Desde el punto de vista del Reconocimiento Sintáctico de Formas se admite que la fase de aprendizaje de los modelos puede ser cubierta por medio de inferencia inductiva. Es sabido que con solo datos positivos ni siquiera es correctamente inferible la clase de los lenguajes regulares. En este papel se propone un algoritmo eficiente que trabaja con datos positivos y negativos y que es capaz de identificar en el límite cualquier lenguaje regular.

1. INTRODUCCION

El paradigma de Inferencia Inductiva (Gold,67) constituye la base de una de las aproximaciones al problema del aprendizaje automático. En este marco se ha producido toda una serie de resultados teóricos destinados a discernir aquello que es aprendible mediante estrategias algorítmicas, así como resultados acerca de las propiedades generales de los algoritmos de inferencia. Por otra parte, en conexión con la disciplina conocida como Reconocimiento Sintáctico de Formas, se ha desarrollado una amplia variedad de algoritmos de inferencia de lenguajes con el propósito de que puedan ser usados en la fase de aprendizaje en diferentes tareas realistas de Reconocimiento de Formas.

Si se pasa revista a los métodos concretos de inferencia que han sido propuestos en los últimos veinte años, inmediatamente llama la atención el hecho

* Trabajo parcialmente subvencionado por CICYT TIC-0448/98

de que en su mayor parte se limitan a la inferencia de lenguajes regulares y siempre son métodos que sólo usan datos positivos. Esta situación es, al menos en apariencia, paradójica. Se sabe (Gold,67) que a partir de presentación positiva no es identificable la clase de los lenguajes regulares; por el contrario, cualquier clase enumerable de lenguajes recursivos es identificable a partir de presentación completa (que contenga datos positivos y negativos). ¿Cual es la razón de esta renuncia a usar la información negativa disponible? En muchas ocasiones la propuesta de un método de inferencia no se acompaña de una discusión del porqué de esta limitación. Cuando tiene lugar una discusión tal, se suele argumentar de manera más bien confusa que el uso de información positiva y negativa convirte en intratable el problema de la inferencia de lenguajes regulares. En tales ocasioones se suele hacer referencia a que el problema de encontrar un AFD con un número mínimo de estados compatible con un conjunto S_+ de datos positivos y un conjunto S_- de datos negativos es NP-duro (Gold,78). La relación entre este resultado y el problema de inferencia de lenguajes regulares tiene su explicación en el hecho de que un algoritmo que con cada entrada de un nuevo dato (positivo o negativo) encontrara un AFD con un número mínimo de estados compatible con los datos ya procesados, posee la propiedad de identificar en el límite cualquier lenguaje regular. Del resultado de Gold se desprende que un método de inferencia como el descrito no podría ser eficiente. Deducir de ello la intratabildad del problema de inferencia de lenguajes regulares a partir de presentación completa es, sin embargo, erroneo. Esta conclusión resulta tanto más sorprendente si se tiene en cuenta que en el mismo papel en que se apoya (Gold,78) se propone un algoritmo polinómico (que obviamente no pasa por encontrar un AFD para cualquier conjunto finito de datos) que permite identificar cualquier lenguaje regular.

En este papel se propone un nuevo algoritmo que posee la propiedad de identificar la clase de los lenguajes regulares a partir de presentación completa. El método produce, dada una muestra arbitraria (S_+, S_-) un AFD (no necesariamente mínimo) compatible con ella. Cuando la muestra (S_+, S_-) cumple ciertas condiciones de representatividad del lenguaje desconocido, el algoritmo produce A(L), el aceptor canónico (AFD mínimo) del lenguaje. Por razones de espacio no incluiremos la demostración que puede encontrarse en (Oncina y Garcia,90).

El método que se presenta es un método de agrupamiento de estados. Es conocido que si S_+ es una muestra estructuralmente completa de un cierto lenguaje L (todas las transiciones en A(L) son usadas en la aceptación de las palabras de

S_+) existe una partición π del conjunto de estados del Arbol Aceptor de Prefijos de S_+, $PT(S_+)$ tal que $PT(S_+)/\pi$ es isomorfo a A(L). Las palabras en Σ^*-L permiten descartar determinadas particiones por lo que el problema de iknferencia puede plantearse como un problema de búsqueda (guiada por S_-) en el reticulo de todas las posibles particiones. Desgraciadamente el espacio de búsqueda crece exponencialmente con la talla del conjunto de estados en $PT(S_+)$ y, por tanto, con la talla de S_+.

En lugar de la búsqueda exhaustiva, el algoritmo que se propone va juntando pares de estados en $PT(S_+)$ según un cierto orden siempre que el autómata resultante rechace todas las palabras en S_-. El algoritmo produce en tiempo polinómico con la talla de (S_+, S_-) un AFD compatible con dichos conjuntos y, ademas, converge al aceptor canónico del lenguaje desconocido.

EL algoritmo comparte con el propuesto por Gold la desventaja de no ser incremental. Recientemente (Feldman,89) se ha demostrado la imposibilidad de cualquier algoritmo incremental que identifique la clase de los lenguajes regulares a partir de presentación completa a menos que los datos se den en orden lexicográfico.

El algoritmo de Gold presenta, desde el punto de vista de su posible aplicación al aprendizaje en RSF, el inconveniente de que no suele generalizar S_+ a menos que dicho conjunto cumpla determinadas condiciones bastante exigentes. En cambio nuestro algoritmo esta libre de tal inconveniente por lo que, en principio, parece más apto para ser usado en tareas de aprendizaje.

2. CONCEPTOS BASICOS Y NOTACION

Σ denota un alfabeto finito y Σ^* denota el conjunto de todas las palabras sobre Σ. $e \epsilon \Sigma^*$ denota la cadena vacía. Si $u \epsilon \Sigma^*$, entonces $|u|$ denota la longitud de u. Si $u = vw$ diremos que v (w) es un **prefijo** (**sufijo**) de u. Un lenguaje sobre Σ es cualquier subconjunto de Σ^*.

Si L es un lenguaje sobre Σ, se define el **conjunto de prefijos de L** como:

$$Pr(L) = \{ u \epsilon \Sigma^* \mid] - v \epsilon \Sigma^*, uv \epsilon L \}$$

y se define el **conjunto de buenos finales de u en L** como:

$$T_L(u) = \{ v \epsilon \Sigma^* \mid uv \epsilon L \}$$

Un lenguaje L es regular si es aceptado por un autómata finito (AF). Un AF se define como $A = (Q, \Sigma, \delta, q_0, F)$ con Q conjunto finito de estados, $q_0 \epsilon Q$ estado inicial, $F \subset Q$ conjunto de estados finales y $\delta: Q \times \Sigma \longrightarrow 2^Q$ una función parcial (2^Q

denota el conjunto de las partes de Q). A es determinista si para todo $q \in Q$ y para todo $a \in \Sigma$, $\delta(q,a)$ contiene a lo sumo un elemento. L(A) denota el lenguaje aceptado por A.

Si $A = (Q,\Sigma,\delta,q_0,F)$ es un AF y π es una partición de Q, $B(q,\pi)$ denota el único bloque de π que contiene a q. El conjunto cociente $\{ B(q,\pi) \mid q \in Q \}$ se denotará Q/π. Dados un AF A y π una partición de Q, se define el autómata cociente A/π como:

$A/\pi = (Q/\pi,\Sigma,\delta',B(q_0,\pi),Q/\pi \cap F)$ con δ' definida:

$\forall B,B' \in Q/\pi$, $\forall a \in \Sigma$, $B' \in \delta(B,a)$ sii $\exists q,q' \in Q$, $q \in B$, $q' \in B'$ tales que $q' \in \delta(q,a)$.

Dados A y π de Q, se tiene que $L(A) \subset L(A/\pi)$.

Dado $A = (Q,\Sigma,\delta,q_0,F)$ $L(A)=L$, la partición π_L definida como $B(q,\pi_L) = B(q',\pi_L)$ sii $\forall x \in \Sigma^*$ $\delta(q,x) \cap F \neq \phi$ sii $\delta(q',x) \cap F \neq \phi$ define un A/π_L que el con un número mínimo de estados que acepta L(A); dicho autómata se denomina aceptor canónico de L y se denota A(L).

$A(L) = (Q,\Sigma,\delta,q_0,F)$ se puede definir a partir de L como:

$Q = \{ T_L(u) \mid u \in Pr(L) \}$; $q_0 = T_L(e)$; $F = \{ T_L(u) \mid u \in L \}$;

$\delta(T_L(u),a) = T_L(ua)$ con $u,ua \in Pr(L)$

Una muestra S de un lenguaje L es un conjunto finito de palabras que se puede expresar como $S = S_+ \cup S_-$ de manera que S_+ es un subconjunto de L (muestra positiva) y S_- es ta incluido en el complementario de L (muestra negativa). Denotaremos una muestra como $S = (S_+,S_-)$.

3. ALGORITMO DE INFERENCIA

Asumiremos el orden lexicográfico en Σ^* que denotaremos por $<$. Puesto que, dada S_+ sobre Σ, los estados en $PT(S_+)$ son los prefijos de las palabras en S_+, es decir $Pr(S_+)$, la relación $<$ puede interpretarse como un buen orden en el conjunto de estados. Dicha relación puede extenderse al conjunto de bloques resultantes de hacer una partición en $Pr(S_+)$.

Definición 3.1 Sea π es una partición de $Pr(S_+)$ y sean dos bloques $B_i,B_j \in \pi$. Diremos que $B_i < B_j$ si y sólo si existe algún $u \in B_i$ tal que para cualquier $v \in B_j$, $u < v$.

Procedimiento Juntar

Dada una partición π en $Pr(S_+)$ y $B_i,B_j \in \pi$ se define el agrupamiento de bloques que denotaremos $J(\pi,B_i,B_j)$ como sigue:

$J(\pi,B_i,B_j) = (B\epsilon\pi \mid B = B_i,\ B = B_j) \cup (B_i \cup B_j)$

En adelante se asumirá que el índice correspondiente a un bloque de una partición coincide con el índice de la menor cadena que contiene dicho bloque. Si un bloque contiene una sóla cadena se representará indistitamente como conjunto de una sóla cadena o como cadena ($B_i = \{u_i\}$ podrá representarse como u_i).

Definición del algoritmo de inferencia

Sea $S = (S_+,S_-)$ y sea $PT(S_+)$ el Arbol Aceptor de Prefijos de S_+. Sea $Pr(S_+) = \{ u_0, u_1, \ldots, u_r \}$ con $u_0 = e$ el conjunto de estados en $PT(S_+)$. Con entrada S el algoritmo de inferencia produce como salida el AFD $A(S,r) = A_0/\pi_r$ con $A_0=PT(S_+)$ y π_r definida como sigue:

Para $n = 0, 1, \ldots, r$, $\pi_n = \pi'_n \cup \{ u_{n+1}, \ldots, u_r \}$, siendo π'_n una partición de $\{ u_0, \ldots, u_n \}$ que se defina recursivamente:

$\pi'_0 := \{ u_0 \}$

si existen dos bloques $B,B'\epsilon\pi'_{n-1}$, $a\epsilon\Sigma$ tales que B' y u_n son a-sucesores de B y B' es el menor a sucesor de B en π'_{n-1} que cumple que $S_- \cap L(A_0/J(\pi'_{n-1},B,u_n)) = \phi$,
entonces $\pi'_n := J(\pi'_{n-1},B,u_n)$

si no si existe $B\epsilon\pi'_{n-1}$ y B es el menor bloque en π'_{n-1} tal que
$S_- \cap L(A_0/J(\pi'_{n-1},B,u_n)) = \phi$,
entonces $\pi'_n := J(\pi'_{n-1},B,u_n)$
si no $\pi'_n := \pi'_{n-1} \cup \{ u_n \}$

Ejemplo 3.1.- Sean $S_+ = \{ 111, 000, 11101, 01 \}$ una muestra positiva y $S_- = \{ 0, 1, 00, 11 \}$ una muestra negativa.

El primer paso consiste en constrir A(S,0) como $A_0=PT(S_+)$ tal como se muestra en la Fig.3.1.

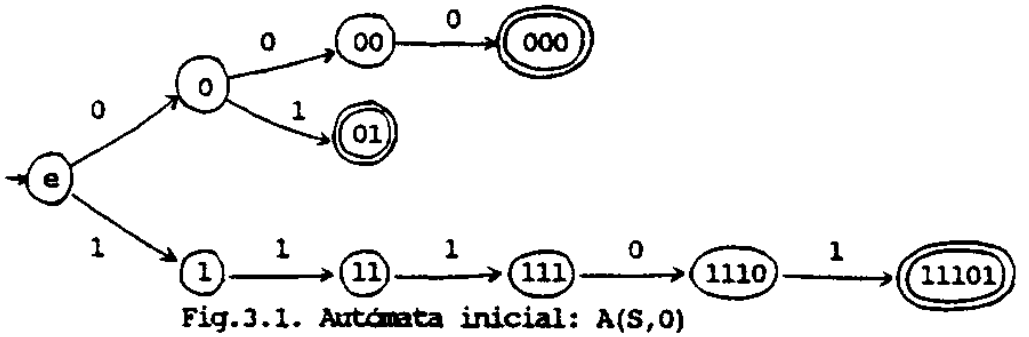

Fig.3.1. Autómata inicial: A(S,0)

El conjunto de prefijos de S_+ en orden lexicográfico es:
< e, 0, 1, 00, 01, 11, 000, 111, 1110, 11101 >

Obtención de A(S,1), (u_1 = 0):

Como e no contiene ningún 0-sucesor que sea menor que 0, se intentará juntar el estado 0 con el estado e, es decir, $J(\pi_0,e,0)$ resultando el autómata $A_0/J(\pi_0,e,0)$ que se muestra en la Fig.3.2. que no es válido puesto que $L(A_0/J(\pi_0,e,0)) \cap S_- = \phi$ (se ve que el dato negativo $1 \in S_-$ es aceptado por el autómata así obtenido.

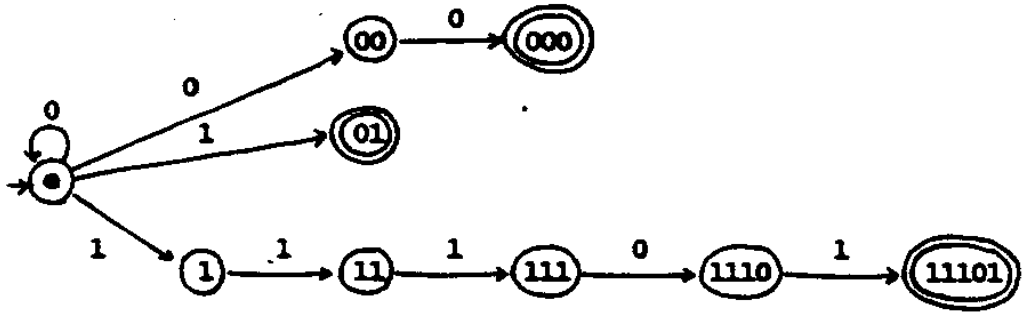

Fig.3.2. $A_0/J(\pi_0,e,0)$ (incompatible con los datos)

Puesto que ya no hay ningún otro estado con el que intentar juntar u_1 = 0, se tiene que A(S,1) = a(S,0).

Obtención de A(S,2), (u_2 = 1)

El algoritmo obtiene $A_0/J(\pi_1,e,1)$ que acepta $11 \in S_-$. Prueba a continuación con $A_0/J(\pi_1,0,1)$ que de nuevo acepta 11. Por tanto, se tiene que A(S,2) = A(S,1), ($\pi_2 = \pi_1$).

Obtención de A(S,3), (u_3 = 00)

$A_0/J(\pi_2,e,00)$ acepta $0 \in S_-$

$A_0/J(\pi_2,0,00)$ acepta $00 \in S_-$

$A_0/J(\pi_2,1,00)$ que se muestra en la Fig 3.3 resulta compatible con los datos. Por tanto A(S,3) = $A_0/J(\pi_2,1,00)$

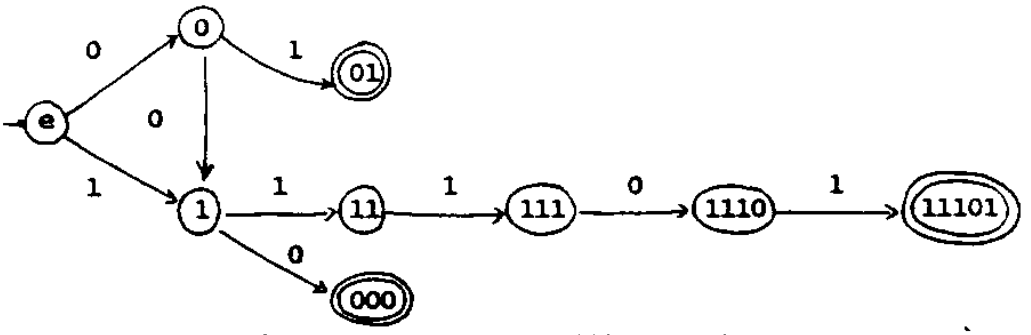

Fig.3.3. A(S,3) = $A_0/J(\pi_2,1,00)$

La partición de los estados explorados es:

π'_3 = { e }, { 0 }, { 1, 00 }

Los sucesivos resultados para A(S,n), n=4,...,9 se muestran en las Fig.3.4 a la Fig. 3.9. El autómata inferido, A(S,9), reconoce el lenguaje sobre { 0,1 } formado por cadenas tales que la diferencia entre el número de 0's y el número de 1's es en valor absoluto un múltiplo de 3.

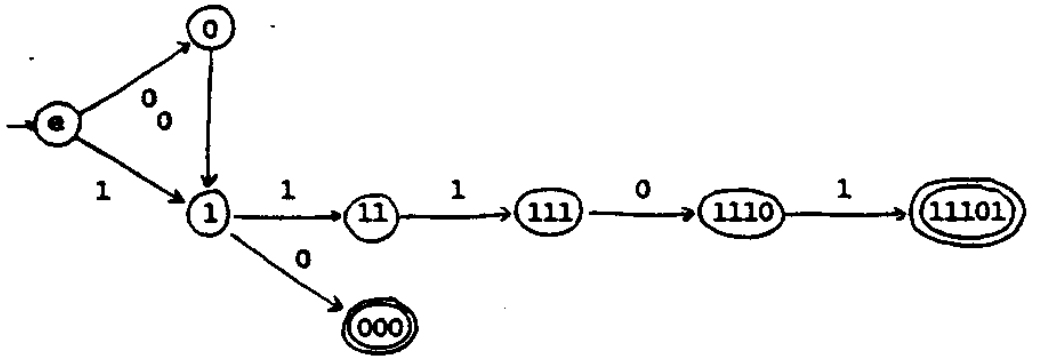

Fig.3.4. $A_0/J(\pi_3,e,01)$

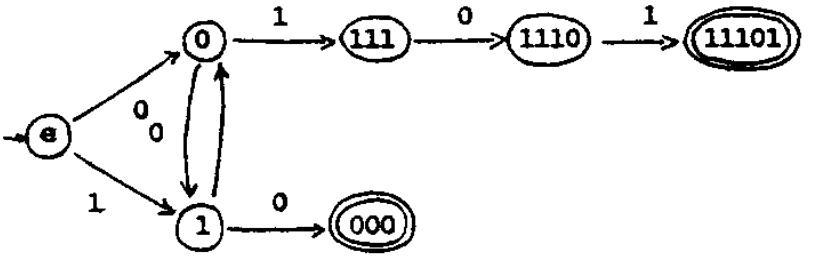

Fig.3.5. $A_0/J(\pi_4,0,11)$

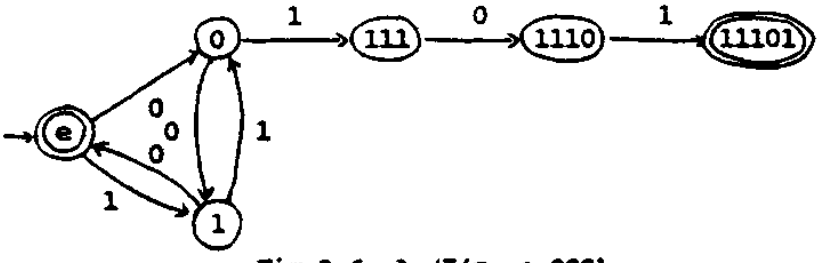

Fig.3.6. $A_0/J(\pi_5,e,000)$

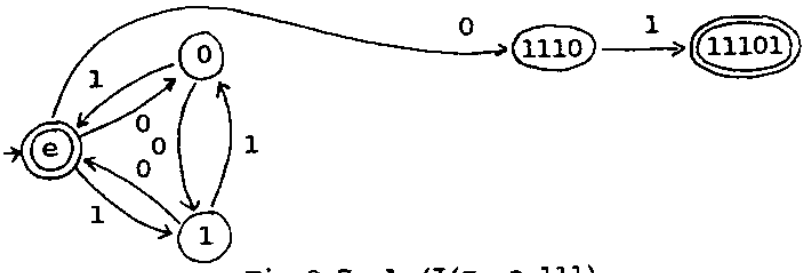

Fig.3.7. $A_0/J(\pi_6,e,111)$

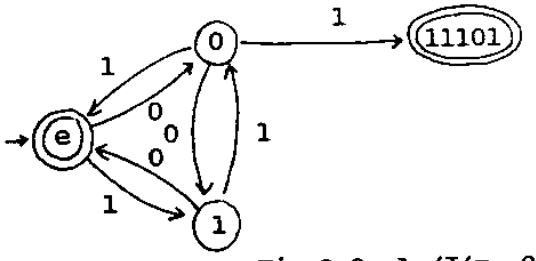

Fig.3.8. $A_0/J(\pi_7,0,1110)$

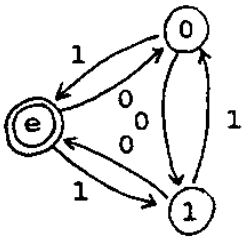

Fig.3.9. $A_0/J(\pi_8,e,11101)$

4. REFERENCIAS

-GOLD,E.M.(1967). "Language Identification in the Limit". Inf. and Control, 10, 447-474.

-GOLD,E.M.(1978). "Complexity of automaton identification from given data". Inf. and Control, 37, 302-320.

-ONCINA,J, and GARCIA,P.(1990). "Un algoritmo polinómico para la inferencia de lenguajes regulares". UPV/DSIC-II 1990/1

-FELDMAN,A. (1989). "Learning Automata from Ordered Examples". Proc. of the 1988 workshop on computational learning theory, MIT. Aug. 1988. D. Haussler & L. Pitt. Eds. Morgan Kaufmann Publishers, Inc. 1989.