Learning CFG using Improved TBL algorithm

Nisha Bhalse[1], Vivek Gupta[2]

[1]Research scholar, Department of Computer Science & Engineering
IES IPSA Indore

[2]Reader, Department of Computer Science & Engineering
Institute of Engineering & Science, IPS Academy
Indore- 452012 , M.P., India

[1]nishabhalse@gmail.com,[2]guptav06@gmail.com

ABSTRACT

Grammar Induction is the process of learning context free grammar from finite set of data of the positive and negative sample of language. The Problem of learning context free grammars from example has two aspects viz. determining the grammatical structure (topology) of the unknown grammar and identifying non-terminals. To solve these problems through a new hypothesis representation method called tabular representation is used , This method uses a table like data structure similar to the parse table used in the Cocke-Younger-Kasami parsing algorithm (CYK Algorithm) for context free grammar of Chomsky Normal Form. In this paper we introduce some improvement to the tabular representation algorithm (TBL) called Improved TBL algorithm (ITBL) dedicated to inference of grammar in Chomsky normal form. This ITBL algorithm uses a Genetic Algorithm (GA) to solve partitioning problem and focuses on the modified fitness function. It gives faster and optimal solutions for 5-8 string length.

KEYWORDS

Grammatical inference, context free grammar, partitioning problem, CYK algorithm, TBL algorithm.

1. Introduction

Induction learning of formal language, often called grammatical inference, is an active research area in machine learning and computational learning theory [1]. The problems of learning finite automata from positive and negative examples have been widely and well studies. Learning of formal languages inherently contains computationally hard problem [2].

In this paper we introduce some improvement to the TBL algorithm that dedicated to inference of CFG in Chomsky Normal Form(CNF).The TBL (tabular representation algorithm) was proposed by Sakakibara and kondo. The TBL algorithm is a novel approach to overcome the difficulties of learning context free grammars from example without structure information available. The algorithm uses a new representation method, called tabular representation, which consist of a table like data structure similar to the parse table used in Cocke –Younger- Kasami parsing algorithm [3]). This tabular representation efficiently represents a hypothesis space of context free grammar.

The tabular representation use the dynamic programming technique to efficiently store the exponential number of all possible grammatical structure in a table of polynomial size. The concept of tabular representation for context free grammar corresponds to the prefix tree automata by employing this representation method, the problem of context free grammar from examples can be reduced to the problem of partitioning the set of non terminals. We use the genetic algorithm for solving this partitioning problem, because the partitioning problems contains NP-hard problem.

The proposed improved TBL algorithm outperform the standard one by providing the approach not so much vulnerable to block size and population size, and is able to find the solution faster.

In section 2 we review CFG learning method based on the genetic algorithm. Next we give a brief introduction to context free grammar induction in section 3,and section 4 which followed by a representation of the tabular CFG induction. Section 5 describe the improved version of TBL algorithm .In section 6 we present result of experiment which show the distinct improvement in performance of algorithm and section 7 in which conclusion and future scope for researchers.

2. Related Work

The state of the art of learning context free grammar is mainly made of negative results. The practical implication of context free language is important. So, research has concentrated essentially in three directions: Studying sub classes of linear languages, learning from structured data and developing heuristics based on genetic algorithms.

Many researchers consider linearity to be a necessary condition for learning to be possible, Successful results were gained over even linear grammars and deterministic linear grammars.

Grammatical inference can be considered as a difficult optimization task. Evolutionary approaches are probabilistic search technique especially suited for search and optimization problems where the problem space is large, complex and contains difficulties. The different representations of grammars was explored in where experimental results showed that an evolutionary algorithm using standard context free grammars (BNF) outperforms those using Chomsky Normal Form (CNF) or bit-string representations. TBL algorithm dedicated to inference of CFG in Chomsky Normal Form(CNF).

In a genetic approach was used for inferring grammars for context free grammar , Apply a GA to solve the partitioning problem for the set of states in the prefix-tree automaton. TBL algorithm used to learn CFGs by introducing the tabular representation. This approach is based on inductive CYK algorithm, incremental learning, and search for rule sets. Another genetic-based system using CYK to parse examples form learning contex-free language is Grammar-based Classifier System (GCS).

3. Context free grammar parsing

3.1 Context free grammar

A context freee grammar (CFG) is quadruple G=(N,∑.P,S), Where ∑ is finite set of non terminal symbols such that ∑∩V=Φ.P is set of production rules in the form W→ ß where W€V and ß€(V U∑).S€V is a special non terminal symbol is called start symbol. All derivation start using the start symbol S.A derivation is sequence of rewriting the string containing symbol (V U∑) using production rule of CFG.

G=(N,∑.P,S), where ∑={a,b},N={A,B},P={A→ a, A→ BA, A→ AB, A→ AA, B→ b},S= A, then the derivation can have the form:

A→AB→ABB→AABB→aABB→aABb→aAbb→aabb.
The derivation ends when there are no more nontermminal symbols left in the string.The language generated by CFG is denoted L(G)={x€Σ^*|S=>$_G{}^*$x},where x represent words of language L. The word x is a string of any number of terminal symbols derived using production rules from CFG G.

A CFG G= (N,∑.P,S), is in Chomsky Normal Form if every production rule in R has form: A →BC or A→a, where A,B,C €V and a €∑.

A CFG describe specific language L(G) and can be used to recognize word S (String) that are members of language(G).But to answer the question which word are member of the language L(G) and which are not we need a parsing algorithm .This problem ,called membership problem can be solved using the CYK algorithm.

3.2 CYK Algorithm

The **Cocke–Younger–Kasami (CYK) algorithm** (alternatively called **CKY**) is a parsing algorithm for context-free grammars. It employs bottom-up parsing and dynamic programming.context-free grammar may be transformed to a CNF grammar expressing the same language .

Algorithm 1

The input is a string *S* consisting of *n* characters: a_1 ... a_n.
The grammar contains *r* non terminal symbols R_1 ... R_r.
This grammar contains the subset R_s which is the set of start symbols.

```
let P[n,n,r] be an array of Booleans. Initialize all elements of P to false.

for each i = 1 to n
  for each unit production R_j -> a_i
    set P[i,1,j] = true
for each i = 2 to n -- Length of span
  for each j = 1 to n-i+1 -- Start of span
  **for each** $k$ = 1 to $i$-1 -- *Partition of span*
    **for each** production $R_A$ -> $R_B$ $R_C$
      **if** $P[j,k,B]$ and $P[j+k,i-k,C]$ **then** set $P[j,i,A]$ = true
**if** any of $P[1,n,x]$ is true ($x$ is iterated over the set $s$, where $s$ are all the indices for $R_s$) **then**
  $S$ is member of language
**else**
  $S$ is not member of language
```

Output: If P[1,n] contain the start symbol S then parsing succeeds.

The CYK algorithm is a polynomial time algorithm to solve the parsing problem of Context free grammars using "dynamic Programming". The CYK algorithm assumes Chomsky normal form of CFGs, and the essence of the algorithm is the construction of a triangle parse table T. This parsing algorithm needs CFG to be in CNF, that is, the right hand side of each rule must expand to either two non terminal or to a single terminal. Restricting a grammar to CNF does not lead to any loss in converting the CFG into a corresponding CNF grammar that accepts exactly the same set of string as the original grammar. It is also possible to extend the CYK algorithm to handle some context free grammars which are not in CNF.

First we introduce the tabular parsing method of the CYK algorithm, and next we propose a tabular representation method which efficiently represents an exponential no. of possible grammatical structure for learning CFGS. The CYK algorithm assumes Chomsky normal form of CFGS, and the essence of the algorithm is the construction of a triangle parse table T. given a CFG $G = (N,\sum.P,S)$, and the input string $W=a_1 a_2 a_3 a_4 \ldots\ldots\ldots\ldots a_n$ in to be parsed according to G, each element of T, denoted $t_{i,j}$ for $1\leq i\leq n$ and $1\leq j\leq n-i+1$, will have a value which is a subset of N.

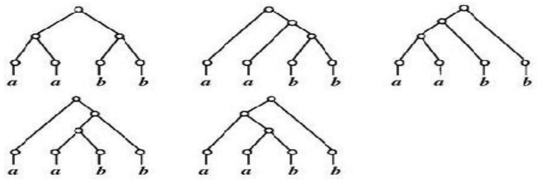


Figure .1 All possible grammatical structure for the string "aabb"

The interpretation of T is that a non terminal A will be in $t_{i,j}$ if and only if $A=>a_i a_{i+1} a_{i+2 \ldots\ldots\ldots} a_{i+j-1}$ that is, A derives the substring of w beginning at position I and of length j. To determine whether the string w is in L(G), the algorithm computes the parse table T and look to see whether S is in entry $t_{1,n}$.

In the first step of constructing the parse table , the CYK algorithm sets $t_{i,1=}\{A| A\text{->}a_i$ is in p}.In the second step, it assumes that $t_{i.j}$ has been computed for $t_{i,j}$ by examining the non terminals in the following pair of entries .

$(t_{i,1}, t_{i+1,j-1}), (t_{i,2}, t_{i+2,j-2} \ldots\ldots\ldots\ldots\ldots (t_{i,j-1}, t_{i+j-1,1})$

And if B is in $t_{i,k}$ and C is in $t_{i+k,j-k}$ for some k($1\leq k<j$) and the production A->BC is in P, adding A to $t_{i,j}$.

For example, we consider a simple CFG G =(N,∑.P,S), of Chomsky normal form where

V={S,A} , Σ={a,b} and

R={ S->AA, S->AS, S->b, A->SA, A->a}

This CFG generates a string "abaaa" that is S-->abaaa, and the parse table T for "abaaa" becomes as follow:

5	S,A				
4	S,A	S,A			
3	S,A	S	S,A	S	
2	S	A	S	S	
j=1	A	S	A	A	A
	i=1	2	3	4	5
	a	b	a	a	a

Figure 2 .The parse table T of G for the string "abaaa".

This parse table can efficiently store all possible parse tree of a G for string " abaaa".

We propose a representation method using this parse table for efficiently representation hypothesis of CFGs.

Given a positive example $w=a_1a_2a_3.....................a_n$, the tabular representation for w is the triangular table T(w) where each element denoted $t_{i,j}$,for $1\leq i\leq n$ and $2\leq j\leq n-i+1$, contain the set $\{X_{i,j,k1}..........\ X_{i,j,k}\}$ of j-1 distinct non terminal .For j=1,$t_{i,1}$ is the singleton set $\{X_{i,1,1}\}$. The primitive CFG G (T(w))= (∑,V.R,S),derived from the tabular representation T(w) is defined as follows.

$V=\{\ X_{i,j,k}|\ 1\leq i\leq n,\ 1\leq j\leq n-i+1,\ 1\leq k<j\}$
$R=\{X_{i,j,k}\text{–>}X_{i,k.l1}\ \ X_{i,k.l2}\ |1\leq i\leq n,\ 1\leq j\leq n-i+1,\ 1\leq k<j,\ 1\leq l_1<k,\ 1\leq l_2<j-k\}\ U\ \{\ X_{i,1,1}\text{–>}a_i|1\leq i\leq n\}U\{\ S\text{–>}X_{i,n,k}|\ 1\leq k\leq n-1\}$

Each non terminal $X_{i,j,k}$ at entry $t_{i,j}$,and the production rules $X_{i,j,k}$ –>$X_{i,k,l1}$ $X_{i+k,j-k,l2}$ with ,the right hand sides of the non terminals at entries $t_{i,k}$ and $t_{i+k,j-k}$.

n	$\{X_{1,n,1}$ ....,$X_{1,n,n1}\}$				
n-1	$\{X_{1,n-1,1}$......,$X_{1,n-1,n-2}\}$	$\{X_{2,n-1,1}$ ....,$X_{2,n-1,n-2}\}$			
⋮	⋮	⋮	……..		
2	$\{X_{1,2,1}\}$	$\{X_{2,2,1}\}$	……..	$\{X_{3,2,1}\}$	
j=1	$\{X_{1,1,1}\}$	$\{X_{2,1,1}\}$	……..	$\{X_{3,1,1}\}$	$\{X_{4,1,1}\}$
	i=1	2	……..	3	4

Figure 3. The tabular representation for w.

The primitive of CFG G(T(w) generate the string w but it can generate all possible grammatical structure of w. For example when a positive example "aabb" is a string and it length is number, the tabular representation T(aabb) and the production G(T(aabb)) are follows:

4	$\{X_{1,4,1}$ $,X_{1,4,2},X_{1,4,3}\}$			
3	$\{X_{1,3,1},X_{1,3,2}\}$	$\{X_{2,3,1}$ $,X_{2,3,2}\}$		
2	$\{X_{1,2,1}\}$	$\{X_{2,2,1}\}$	$\{X_{3,2,1}\}$	
j=1	$\{X_{1,1,1}\}$	$\{X_{2,1,1}\}$	$\{X_{3,1,1}\}$	$\{X_{4,1,1}\}$
	i=1	2	3	4
	a	a	b	b

P={ $S \rightarrow X_{1,4,1}$, $S \rightarrow X_{1,4,2}$, $S \rightarrow X_{1,4,3}$,
$X_{1,4,1} \rightarrow X_{1,1,1}\ X_{2,3,1}$, $X_{1,4,1} \rightarrow X_{1,1,1}\ X_{2,3,2}$, $X_{1,4,2} \rightarrow X_{1,2,1}\ X_{3,2,1}$,
$X_{1,4,2} \rightarrow X_{1,2,1}\ X_{3,2,1}$, $X_{1,4,3} \rightarrow X_{1,3,2}\ X_{4,1,1}$, $X_{1,3,1} \rightarrow$
$X_{1,1,1}\ X_{2,2,1}$,
$X_{1,3,2} \rightarrow X_{1,2,1}\ X_{3,1,1}$, $X_{2,3,1} \rightarrow X_{2,1,1}\ X_{3,2,1}$, $X_{2,3,2} \rightarrow X_{2,2,1}\ X_{4,1,1}$,
$X_{1,2,1} \rightarrow X_{1,1,1}\ X_{2,1,1}$, $X_{2,2,1} \rightarrow X_{2,1,1}\ X_{3,1,1}$, $X_{3,2,1} \rightarrow X_{3,1,1}\ X_{4,1,1}$,
$X_{1,1,1} \rightarrow a$, $X_{2,1,1} \rightarrow a$, $X_{3,1,1} \rightarrow b$, $X_{4,1,1} \rightarrow b$ }

Figure 4. The tabular representation T(aabb) and the derived primitive CFG G(T(aabb)).

The size of G(T(aabb)) is polynomial of the length and CFG G(T(aabb)) can generate all possible grammatical structure for aabb as shown in figure.

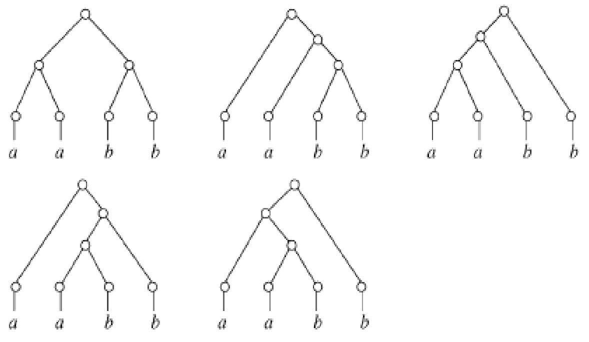


Figure 5. All possible grammatical structure of the string aabb

4. CFG induction using TBL algorithm

Grammar induction target is to find a CFG structure (Set or rules & the set of non terminals) using positive and negative example in a form of words .It is very complex problem, especially hard is finding grammar topology (set of rules).The complexity results from great amount of possible grammatical structure that grow exponentially with length of the string. The TBL algorithm combines all the grammar from the created tabular representation, by reducing the number of non terminal distinct ones and finding the smallest grammar from the learning set is most difficult part of the learning task. This is the NP- hard problem and sakakibara uses genetic algorithm with selection, structure crossover ,structure mutation and a special mutation then deletes randomly selected element with a small probability.

4.1 TBL Algorithm

TBL algorithm input:

U is the learning set ,$U=U_+ \cup U-$.
U_+ is the set of positive examples, whereas U- is the set of negative examples.

4	<12><13><14>			
3	<8><9>	<10><11>		
2	<5>	<6>	<7>	
j=1	<1>	<2>	<3>	<4>
	i=1	2	3	4
	a	a	b	b

Figure. 6. Tabular representation of the String aabb.

1. Create tabular representation T (w) of each positive example w in U_+ .
2. Derive primitive CFG G(T(w)) and G^w (T (w)) for each w in U_+.
3. Create union of all primitive CFG's, $G(T(U_+)) = U_{w\in U}{}^{+}G^{W}(T(w))$.
4. Find smallest partition Π_* such that $G(T(U^+))/\pi_p$ is coherent with U , which is compatible with all positive and negative examples U + and U-
5. Return result CFG $G(T(U^+))/\pi_p$.

4.2 Partitioning the set of non-terminals

Partitioning the set of non-terminals is an example of a well known set partition problem ,which is concerned with partitioning a collection of sets into P sub collection such that the maximum number of elements among all the sub collection is minimized. The problem is NP-hard, even when N=2 and the cardinality of each set is 2 .If π is a partition of N ,then for any element $x \in N$ there is a unique element of π containing x denoted $K(x, \pi)$ and called block of Π containing x.
Let π be a partition of the set of V non-terminals of CFG $G=(N,\sum.P,S)$, The CFG $G/\pi=((N',\sum.P',S')$, induced by π from G is defined as follow:

$N' = \pi$,

$P'=\{K(A, \pi) \rightarrow K(B, \pi)\ K(C, \pi) \rightarrow BC \in P\} \cup \{ K(A, \pi) \rightarrow a|A \rightarrow a \in P\}$,

$S'=K(S, \pi)$

TBL uses a genetic algorithm to solve the set partition problem .The use of GA in minimal partition problem .

4.3 Genetic Algorithm

The genetic algorithm (GA) to solve the partitioning problem for the set of non-terminals in the primitive CFGs. The genetic algorithm is a parallel search technique in which a set ,namely the population, of potential solution ,called individual is progressed updated through a selection mechanism and genetic operation ,that is ,the crossover and the mutation .Each individual is coded as a fixed length string ,called a chromosome.

Starting from a randomly generated initial population, each individual is evaluated by the fitness function to be optimized. The population of the next generation is obtained in three steps. Firstly a random selection with repetition is performed, usually on the whole population. The probability of a given individual to be selected is obtained from the ratio between its fitness value and the average fitness value computed over the whole population. Secondly, at a given crossover rate pairs are selected from this temporary population. Each pair of parents give rise to two children which are constructed by taking alternative subparts from their parent chromosomes. After the crossover operation ,the parents are replaced by their children in the temporary population. Thirdly ,some individual are selected at a given at a given mutation rate. One position in the associated chromosomes of each selected individual is randomly chosen and corresponding value is replaced. The final population of the next generation is made of the individuals obtained after fitness proportionate reproduction, crossover and mutation .this process is iterated until some termination criterion is satisfied.

Genetic algorithm have been well studied for the partitioning problem and we take a successful approach by Von Laszewski for partitioning n element into k categories .This approach encodes partitions using group number encoding, that is ,partitions are represented as strings of integer number of length n,

p= $(i_1, i_2, i_3, i_4 .... i_n)$

Where i_j€{1,2,3,……………k} indicates the block number assigne to element j.

For example ,string p=(1 3 4 3 6 3 3 4 5 2 6 4) represent partition

$\Pi_{P:}$ {1}, {10 }, {2 4 6 7}, {3 8 12 }, {9}, {5 11}, and $|\pi_P|$=6.

Group number encoding has own genetic operators: structural crossover and structure mutation.

Structural crossover- This operator take a union of both parents partitions of a randomly selected block. Let's assume we have two individuals p1 and p2 .

p_1=(1 1 2 3 4 1 2 3 2 2 5 3) , encoded partition

π_{P1}: {1 2 6 }, {3 7 9 10}, {4 8 12 }, {5}, {11} and

p_2=(1 1 2 4 2 3 1 2 2 3 3 5) encoded partition

π_{P2}: {1 2 7}, {3 5 8 9}, {6 10 11 }, {4}, {12}.

During structural crossover a random block is selected from one partition, say block 2 from p1 , and is merged with block from p2 . The result is:

p'$_2$=(1 1 2 4 2 3 2 2 2 2 3 5), encoded partition
$\pi_{P'2}$: {1 2}, {3 5 7 8 9 10 }, {6 11 }, {4}, {12 }.

Structure mutation-This operator replaces one integer in string with another string.

Let's assume we have individual *p*2 , that should be mutated:

After replacing single random integer, for example integer number 9 the result is:

p'$_2$ =(1 1 2 4 2 3 2 2 5 2 3 5) , encoded partition

$\pi_{P''2}$: {1 2}, {3 5 7 8 10 }, {6 11 }, {4}, {9 12}.

Special deletion operator → This operator replaces single integer in string with a special value (-1). This value means that this nonterminal is no longer used. This operator is important because it reduces the number of non-terminals in result grammar.

Fitness function.—> Fitness function is a number from interval [0, 1]. The fitness function was designed to both maximize number of accepted positive examples f$_1$ and minimize number of non-terminals f$_2$. If grammar based on individual accepts any negative example it gets fitness 0 to be immediately discarded. C1 and C2 are some constants that can be selected experimentally.

$$f_1(p)= \frac{|\{w \in U_+ | G(T(U_+))/\pi_p\})}{|U_+|} \qquad (1)$$

$$f_2(p)= \frac{1}{|\pi_p|} \qquad (2)$$

$$f(p)=\begin{cases} 0 & \text{if any negative example from } U_- \text{ is generated by } G(T(U_+))/\pi_p \\ \dfrac{C_1 . f_1(p)+C_2 .f_2(p)}{C_1+C_2} & \text{otherwise} \end{cases} \qquad (3)$$

5. Improved TBL algorithm

TBL algorithm is effective but some problem cause during experiment. The main problem was the fitness function, which was not always precious enough to describe difference between individuals. When partitions with small block were used (average size 1-5) the algorithm find a solution ,but when partition with larger blocks were used ,TBL algorithm could not find solution because all individual in th population had fitness zero and started to work completely randomly. To solve this problem we proposed some modification in fitness function.

The proposed improvement affects only the genetic algorithm solving the partitioning problem. Firstly we want to explain the concept of partition block size in initial population partition used by GA group integers that represent non-terminals in blocks. For example, the partition π_p contain of 6 blocks of size 1-4.

π_p={1}, {10},{2 4 6 7}, {3 8 12}, {9},{5 11}

The range of partition block size in initial population has influence on the evolution process. If block are small (1-4) then it is easier to find solution (partition) π generating CFG $G(T(U^+))/\pi$

which do not parse any negative example .on the other hand creating partition that contain bigger blocks should assure that the reduction on non terminal number in $G(T(U^+))/\pi$ (no. of blocks in π) to the optimal level proceeds quicker .(this is slower process than finding solution that do not accept any negative example),because partition in initial generation contain less blocks.

The main problem with using bigger block is low percentage of successful runs if a standard fitness function is used. Usually in such case all individuals (partitions creates CFGs that accept atleast one negative example and then whole population has the same fitness equals zero.To overcomes those difficulties, we have used some modification in GA listed Below.

Initial population block size manipulation :-> During the experiments have tested the influence of block size on the evolution process. Block size in a range of possible size (for example 1-5) ,not one number.

Block delete specialized operator :-> This operator is similar to special deletion operator but removing one non terminal from partition it deletes all non-terminals that are in same randomly selected block.

Modified fitness function :-> To solve problems with using standard fitness function when block sizes are bigger (all individuals have fitness) .We propose modified fitness function that does not give all individuals that accept a negative example the same fitness value. To be able to distinguish between the better and worst solution that still accept any negative example we give that solution negative fitness value(value in range [-1,0]). This fitness function range [-1,1] is used only to be able to compare modified fitness function with standard fitness function more easily but it can be normalized to range [0,1].

$$f_1(p)= \frac{|\{w \in U_+ | G(T(U_+))/\pi_p\})}{|U_+|} \qquad (4)$$

$$f_2(p)= \frac{1}{|\pi_p|} \qquad (5)$$

$$f_3(p)= \frac{|\{w \in U_- | G(T(U_-))/\pi_p\})}{|U_-|} \qquad (6)$$

$$f(p)= \begin{cases} -(C_3.f_3+ C_4.(1-f_1) & \text{if any example from U- is generated by } G(T(U_+))/\pi_p \\ \frac{C_1 . f_1(p)+ C_2 .f_2(p)}{C_1+C_2} & \text{otherwise ,} \end{cases} \qquad (7)$$

The aim of modified fitness function f(p) is to find Context free grammar consistent with a whole learning set. The proposed formula extend eq. 3 by the function $f_3(p)$ used to estimated a fitness function if any negative example is generated by $G(T(U_+))/\pi_p$ .The standard fitness function ,formula (7) does not punish the inferred grammar for generating even one negative sentence but count sum of the normalized number of generated negative examples(formula $C_3.f_3(p)$) and complement of the normalized number of non accepted negative example($C4.(1-f_1(p)$)). Introduce the C_3 and C_4 contant (set to.5) allow to normalized number $f_1(p)$ and $f_3(p)$ on fitness function in a case of negative example. New fitness function uses the same formula $f_2(p)$ for minimize the no. of non-terminals that have used in the sakakibara's formula.

The $C_{1,}C_2$ value is 0.5 in positive example ($f_1(p)$+ $f_3(p)$). The value of $f_1(p)=1$ (the induced grammar which accept all positive and reject all negative example).The fitness function is able to reach the value of 1 only if $f_2(p)=1$.

6. Experiment Result

The experiment results show also the population size has a big influence on the number of successful runs when the improved fitness function is used. Successful runs values (Succ. runs) are defined as (successful run number/overall run number) .100[%]. Average generations (Avg. gen.) indicate the average number of generations needed to reach 100% fitness.

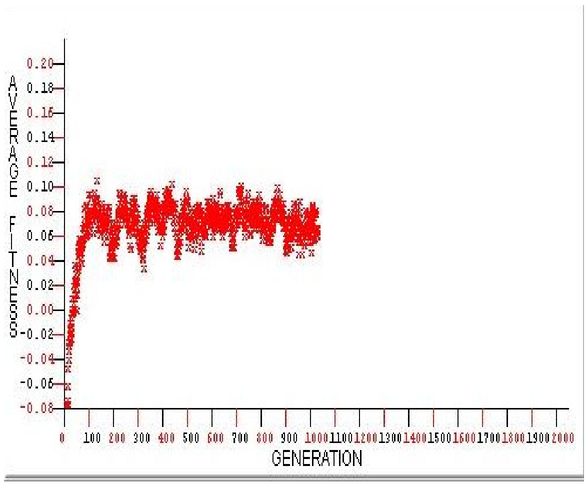


Fig 7(a). Learning over examples from language L = { a ^{n} } Initial population block size: 1-6 .

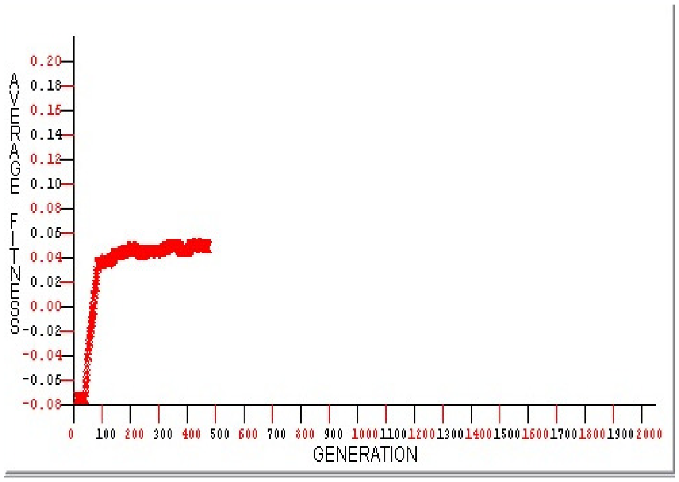


Fig 7(b). Learning over examples from language L = { $a^n b^n$ } Initial population block size: 1-6

In experiments the size of the population increase the value of fitness function gives the greater fitness. The experiment results proved also that the population size has no bigger influence on algorithm effectiveness if the modified fitness function is used. The use of larger blocks improves also the quality of solutions and algorithm speed. Fig. 7 shows more detailed results of experiment.

The solutions are not only better in comparison with standard fitness using larger blocks (2–6), but also better than solutions gained when smaller blocks were used (1–5). Using larger block size reduces also the average number of generations needed to find the best solution. This is possible because each partition π_p in initial population contains smaller number of blocks and CFG $G(T(U_+))/\pi_p$ based on π_p contains less non-terminals. The algorithm finds solution faster because non-terminal number reduction process is quicker (smaller initial number of non-terminals).

7. Conclusion

In this paper we study a new hypothesis representation method, called the tabular representation, based on the parse table for learning CFGs and it is efficiently store the exponential number of all possible grammatical structures in a table of polynomial size. The problem of learning CFGs is reduced to the problem of partitioning the set of non-terminals, and we have used a genetic algorithm for solving the partitioning problem and standard TBL algorithm is its susceptibility to a partition block size in an initial population and also population size. The experiments we have performed showed that for larger blocks the algorithm is not effective, and for small populations. Our work was motivated by Standard TBL algorithm and extends his method to learning CFGs by introducing the

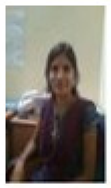
improved tabular representation and some modifications to TBL algorithm which concentrate on genetic algorithm. We have proposed a modified fitness function that distinguishes also individuals accepting negative examples, in contrary to the standard one giving them the same fitness value (value of 0). The proposed fitness function ranges from -1 to 1 in order to compare with standard one, but it can be normalized to range [0,1]. Next, we have introduced a block delete specialized operator, which works similarly to a special delete operator described in but instead of removing one nonterminal from partition it deletes all non-terminals that are placed in some randomly selected block. The aim of this operator is to speed up the process of searching minimal partition. The set of experiments have been performed to determine an improved tabular representation efficiently.

Reference

[1] Y. Sakakibara, Recent advances of grammatical inference, Theoret. Computer Sci. 185 (1997) 15–45.
[2] E.M. Gold, Complexity of automaton identification from given data, Inform. Control 37 (1978) 302–320.
[3] A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation and Compiling, vol. I: Parsing, Prentice-Hall, Englewood Cliffs, NJ, 1972.
[4] Y. Sakakibara, Learning context-free grammars using tabular representations, Pattern Recognition 38 (2005) 1372–1383.
[5] Olgierd Unold, Marcin jaworski : Learning context free grammar using improved tabular representations, Applied Soft Computing 10(2010)44-52.
[6] Y.Sakakibara: Learning context free grammar using tabular representations,Pattern Recognition 38(2005)
[7] Y.Sakakibara, M.Kondo : GA- based learning context free grammar using tabular representations,in: Proceeding of 16th International Conference in Machine Learning(ICML-99)
[8] Kasami T.: An efficient recognition and syntax-analysis algorithm for context-free languages, Sci. Rep. AFCRL-65-558, Air Force Cambridge Research Laboratory, Bedford, Mass. (1965).

[9] Lee L.: Learning of Context-Free Languages: A Survey of the Literature, Report TR-12-96, Harvard University, Cambridge, Massachusetts (1996).
[10] T. Koshiba, E. Ma´ kinen, Y. Takada, Inferring pure context-free languages from positive data, Acta Cybernetica 14 (3) (2000) 469–477.
[11] Y. Sakakibara, Efficient learning of context-free grammars from positive structural examples, Information and Computation 97 (1992) 23–60.
[12] O. Unold, L. Cielecki, Learning context-free grammars from partially structured examples: Juxtaposition of GCS with TBL, in: 7th International Conference on Hybrid Intelligent Systems, IEEE Computer Society Press, Germany, 2007.
[13] P. Wyard, Representational Issues for Context Free Grammar Induction Using Genetic Algorithms, Springer-Verlag, 1994
[14] D.H. Younger, Recognition and parsing of context-free languages in time n3, Information and Control 10 (2) (1967) 189–208.
[15] Higuera C.: Current Trends in Grammatical Inference, In: Ferri F. J. et al. (eds) Advances in Pattern Recognition. Joint IAPR International Workshops SSPR+SPR'2000, LNCS 1876, Springer, Berlin, (2000) 28-31.
[16] Michalewicz Z.: Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin (1996).
[17] Tanaka E.: Theoretical Aspects of Syntactic Pattern Recognition, Pattern Recognition, 28(7): (1995) 1053-1061.
[18] Unold O.: Context-free grammar induction with grammar-based classifier system, Archives of Control Science, 15 (LI) 4: (2005) 681-690.

Authors



Ms. Nisha Bhalse, received B.E. Degree in Computer Science and Engineering from G.E.C.U. Ujjain ,M.P., India and Research Scholar of M.E. Computer Science and Engineering ,IES IPS
Academy Indore, M.P., India. She has presented 1 paper in national conferences.

Mr. Vivek Gupta, Reader, IES, IPS Academy Indore, M.P., INDIA .Received B.E degree in Electrical Engineering from S.G.S.I.T.S Indore ,M.P.,India, in 1996, and the M.E degree in Computer Engineering from S.G.S.I.T.S Indore ,M.P., in 2009. His Research area related to Language learning and language inference. He has presented 1 paper in national conferences. He is Branch Counselor of Computer Society of India (CSI) in CS, IES IPS Ac ademy Indore and he is associate member of CSI and life member of Indian Society for Technical Education (ISTE).