Learning Fuzzy Context-Free Grammar—A Preliminary Report

Olgierd Unold

Member, IEEE
Institute of Computer Engineering, Control and Robotics,
Wroclaw University of Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
olgierd.unold@pwr.wroc.pl
http://olgierd.unold.staff.iiar.pwr.wroc.pl/

Abstract. This paper takes up the topic of a task of learning fuzzy context-free grammar from data. The induction process is divided into two phases: first the generic grammar is derived from the positive sentences, next the membership grades are assigned to the productions taking into account the occurrences of productions in a learning set. The problem of predicting the location of promoters in *Escherichia coli* is examined. Language of bacterial sequence can be described using formal system such as context-free grammar, and problem of promoter region recognition can be replaced by grammar induction. The induced fuzzy grammar was compared to other machine learning methods.

Keywords: Grammatical Inference, Fuzzy Grammar.

1 Introduction

Fuzzy languages and grammars have been introduced in [1]. The fuzzy language theory enables us—contrary to the crisp language theory—to distinct *huge* or *tiny* errors we allow in the input of the parser or the recognizer. Fuzzifying context-free languages (CFLs) is a great step towards a robustness in parsing CFLs. We refer the reader to [2] for more details of fuzzy languages and fuzzy automata.

In this paper, we are interested in inducing a fuzzy context-free grammar (FCFG) that accepts a CFL given a finite number of positive and negative examples drawn from that language. Relatively few efforts have been made to learn FCFGs or fuzzy finite automata that recognize FCFG [3,4,5,6]. This paper addresses a fuzzy context-free grammar induction using a novel flexible approach based on a learning set. First, the algorithm produces the crisp and generic context-free grammar in Chomsky Normal Form (CNF). The generic grammar includes all possible production rules for a choosen learning string, i.e. we assume that in each position of the string we can insert all terminals. Next, the algorithm determines the membership grades for all productions of grammar.

A fuzzy formal language is a formal language where each word has a degree of membership to the language. A FCFG $G = (V, T, P, S, \omega, \otimes, \oplus)$ consists of a set

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 309–312, 2010.

of variables V, a set of terminals T, a set of productions P, start symbol S, ω a set of weights defined over the production rules P, $\otimes$ denotes a t-norm, and $\oplus$, a t-conorm. Productions are of the form $A \xrightarrow{\omega} \alpha$ where $A \in V$, $\alpha \in (V \cup T)$ and $\omega \in [0, 1]$. The empty word is denoted by λ. The fuzzy language $L(G)$ generated by this fuzzy grammar is $\left\{(w, \mu_L(w)) | w \in T^*, S \xrightarrow{*} w\right\}$, $\mu_L(w)$ represents the degree of membership of the word w to the language L and is obtained by apllying the t-norm $\otimes$ to the weights of all productions involved in the generation of w. Should the grammar be ambiguous, and a word w be rechable from S by different sequences of productions, then t-conorm $\oplus$ will be used to calculate the final degree of membership from the degrees of membership obtained through different sequences of productions. A λ-free (fuzzy) context-free grammar G is in CNF iff $P \subseteq V \times [0, 1] \times (T \cup V \times V)$.

2 Learning Fuzzy Grammar

For assigning a (crisp) grammar to a set of learning set, we adopt the algorithm proposed in [7]. All sentences in the learning set are assumed to be of equal length, and for the one choosen sentence (positive) the generic grammar is derived. For example, for a string with length equal to 4 following productions P are obtained:

$S \to AW_1 \quad W_1 \to AW_2 \quad W_2 \to AW_3 \quad W_3 \to A \quad A \to a$

$S \to CW_1 \quad W_1 \to CW_2 \quad W_2 \to CW_3 \quad W_3 \to C \quad C \to c$

$S \to GW_1 \quad W_1 \to GW_2 \quad W_2 \to GW_3 \quad W_3 \to G \quad G \to g$

$S \to TW_1 \quad W_1 \to TW_2 \quad W_2 \to TW_3 \quad W_3 \to T \quad T \to t$

After the generic grammar has been generated, the membership grades are assigned to the production rules. The initial membership grades are set to 0.5. Note that setting the initial values is necessary in order to use different t-norms (like fuzzy AND). The membership grade of each production P_i is calculated as $\mu_{P_i} = \frac{NP_i + NN_i}{PS + NS}$ where PS denotes the number of the positive sentences in the learning set, NS—the number of the negative sentences, NP_i—the number of occurrences of the production P_i in a derivation of the positive sentences, NN_i—the number of non-occurrences of the production P_i in a derivation of the negative sentences (i.e. NN_i is counted for the production $W_i \to xW_{i+1}$ as the sum of the occurrences of the productions $W_i \to yW_{i+1}$, where $x, y \in T$ and $x \neq y$). During a phase of testing, the final degree of membership of each sentence is worked out from the degrees of membership obtained through different sequences of productions. The average function was used. The threshold was set to 0.5, and each sentence with a membership over this threshold is counted as a positive sentence.

In this paper, we address the problem of predicting the location of promoters in *Escherichia coli* [8]. The language of bacterial sequence can be described by using a formal system such as context-free grammar, and problem of promoter region recognition can be replaced by grammar induction. The gene content of these genomes was mostly computationally recognized. However, the promoter regions are still undetermined in most cases and the software able to accurately

Table 1. Comparison of the induced fuzzy grammar (IFG) with different methods. Leung et al. [13] introduced Basic Gene Grammars (BGG) to represent many formulations of the knowledge of *E.coli* promoters. BGG is able to represent knowledge acquired from knowledge-based artificial neural network learning (KBANN approach in [14]), and a combination of grammar of weight matrices [15] and KBANN (denoted as WANN). The development of BGG is supported by DNA-ChartParser. GCS, introduced by O.Unold [16] is a kind of a learning classifier system which evolves using genetic algorithm a population of context-free grammar productions. After each execution four numbers were calculated: True Positives (correctly recognized positive examples), True Negatives (correctly recognized negatives), False Negatives (positives recognized as negatives), and False Positives (negatives recognized as positives). Then the average of these numbers was found and the following measures were calculated: Specificity, Sensitivity, and Accuracy. Specificity is a measure of the incidence of the negative results in testing all the non-promoter sequences, i.e., (True Negatives/(False Positives + True Negatives)) x 100. Sensitivity is a measure of the incidence of positive results in testing all the promoter sequences, i.e., (True Positives/(True Positives + False Negatives)) x 100. Accuracy is measured by the number of correct results, the sum of true positives and true negatives, in relation to the number of tests carried out, i.e., ((True Positives + True Negatives)/Total) x 100.

Method	Specifity	Sensitivity	Accuracy
KBANN	97	16	56
WANN	82	69	75
GCS	94	61	78
IFG	72	78	75

predict promoters in sequenced genomes is not yet available in public domain. Promoter recognition, the computational task of finding the promoter regions on a DNA sequence, is very important for defining the transcription units responsible for specific pathways. A promoter enables the initiation of a gene expression after binding with an enzyme called RNA polymerase, which moves bidirectionally in searching for a promoter, and starts making RNA according to the DNA sequence at the transcription initiation site, following the promoter [9].

The genome can be treated as a string composed of letters A, C, T, G. The goal is, given an arbitrary potential promoter region, to be able to find out whether it is a true or false promoter region. As the learning set the database introduced by M. Noordewier and J. Shavlik to UCI repository was used [10]. The database consists of 53 positive instances and 53 negative instances, 57 letters each. Negative learning sentences were derived from *E. coli* bacteriophage T7 believed to not contain any promoter sites. In order to get an estimate of how well the algorithm learned the concept of promoter, the test set consisting of unseen 36 instances including 18 positive and 18 negative examples was prepared. Positive test instances were prepared by mutating the bases of the randomly chosen positive learning sentences in non-critical positions, negative test instances by mutating in any positions of randomly chosen negative learning sentences. This method increases the amount of available examples and was first proposed in [11]. The induced fuzzy grammar (IFG) achieved 75% accuracy, 72% specificity,

and 75% sensitivity in the testing set. Table 1 compares the results of IFG and three formal system based methods presented in [12]. The results obtained by induced fuzzy grammar are somehow comparable to those methods: Specifity has the lowest value of the compared mathods, but Sensitivity the highest one. Note that replacing symbols A, C, T, G by a, c, t, g in the grammar, one gets a equivalent regular grammar. Moreover, the induced grammar is not ambiguous. However we belive that the use of fuzzy grammars can be a significant step towards a robustness in parsing formal languages, and the proposed approach is flexible enough to deal with complex tasks. The use of different t-norms and t-conorms will be a subject to further testing.

References

1. Lee, E.T., Zadeh, L.A.: Note on fuzzy languages. Inform. Sci. 1, 421–434 (1969)
2. Mordeson, J.N., Mailk, D.S.: Fuzzy Automata and Languages: Theory and Applications. Chapman and Hall, Boca Raton (2002)
3. Mozhiwen, W.: An Evolution Strategy for the Induction of Fuzzy Finite-state Automata. Journal of Mathematics and Statistics 2(2), 386–390 (2006)
4. Wen, M.Z., Min, W.: Fuzzy Automata Induction using Construction Method. Journal of Mathematics and Statistics 2(2), 395–400 (2006)
5. Molina-Lozano, H., Vallejo-Clemente, E.E., Morett-Sanchez, J.E.: DNA sequence analysis using fuzzy grammars. In: IEEE International Conference on Fuzzy Systems, pp. 1915–1921 (2008)
6. Carter, P., Kremer, S.C.: Fuzzy Grammar Induction from Large Corpora. In: IEEE International Conference on Fuzzy Systems (2006)
7. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (2001)
8. Blattner, F., Plunkett, G., Bloch, C., Perna, N., Burland, V., Riley, M., Collado-Vides, J., Glasner, J., Rode, C., Mayhew, G., et al.: The complete genome sequence of Escherichia coli k-12. Science 277, 1453–1462 (1997)
9. Lewin, B.: Genes VII. Oxford University Press, Oxford (2000)
10. Murphy, P.M., Aha, D.W.: UCI Repository of Machine Learning Databases, Department of Information and Computer Science. University of California, Irvine, CA (1992)
11. O'Neill, M.: Escherichia coli promoters: neural networks develop distinct descriptions in learning to search for promoters of different spacing classes. Nucleic Acids Res. 20, 3471–3477 (1992)
12. Unold, O.: Grammar-Based Classifier System for Recognition of Promoter Regions. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007, Part I. LNCS, vol. 4431, pp. 798–805. Springer, Heidelberg (2007)
13. Leung, S.W., Mellish, C., Robertson, D.: Basic gene grammars and dna-chart parser for language processing of Escherichia coli promoter dna sequences. Bioinformatics 17, 226–236 (2001)
14. Towell, G., Shavlik, J.: Extracting refined rules from knowledge-based neural networks. Machine Learning 13, 71–101 (1993)
15. Rice, P., Elliston, K., Gribskov, M.: DNA. In: Girbskov, M., Devereux, J. (eds.) Sequence Analysis Primer, ch. 1, pp. 1–59. Stockton Press (1991)
16. Unold, O.: Context-free grammar induction with grammar-based classifier system. Archives of Control Science 15 (LI) 4, 681–690 (2005)