Vol.2.Issue.1.2014

BULLETIN OF MATHEMATICS AND STATISTICS RESEARCH

A Peer Reviewed International Research Journal

http://www.bomsr.com

ISSN:2348-0580

RESEARCH ARTICLE

SYNTACTIC APPROACHES IN PATTERN RECOGNITIOIN

NUNNA SRINIVASA RAO[1], N.V. RAMANA MURTHY[2]

[1]Departmentof Statistics, [2]Department of Mathematics

Andhra Loyola College (Autonomous),

Vijayawada-520 008, mail id:nunnasr@gmail.com

Author for Correspondence
NUNNA SRINIVASA RAO

Article Info:
Article received :10/02/2013
Revised on:10/03/2014
Accepted on:22/03/2014

ABSTRACT

Linear or piece –wise linear classifiers, Fuzzy, Neural nets are the most common choices in pattern recognition. Here an attempt is made to review various syntactic procedures for pattern recognition.

1. INTRODUCTION

One could possibly distinguish between mathematical Pattern Recognition (primarily cluster analysis) and nonmathematical Pattern Recognition. One of the major differences between these two areas is that the former is far more context dependent than the latter: a heuristic computer program that is able to select features of chromosomal abnormalities according to a physician's experience will have little use for the selection of wheat fields from a photo-interpretation viewpoint. By contrast to this example, a well-designed cluster algorithm will be applicable to a large variety of problems from many different areas. The problems will again be different in structural Pattern Recognition when for instance, handwritten H's shall be distinguished from handwritten A's and so on.

Verhagen (1975) presents a survey of definitions of Pattern Recognition, which also cites the difficulties of an attempt to define this area properly. Bezdek (1981) defines Pattern Recognition simply as *A search for structure in data.*

The most effective search procedure in those instances in which it is applicable is still the *eyeball* technique applied by human *searchers*. Their limitations, however, are strong in some directions: Whenever the human senses, especially the vision, are not able to recognize data or features, the *eyeball* technique cannot be applied.

One of the advantages of human search techniques is the ability to recognize and classify patterns in a non-dichotomous way. One way to initiate this strength is the development of statistical methods in mathematical Pattern Recognition, which in connection with high-speed computers have shown very impressive results. There are data structures, however, that are not

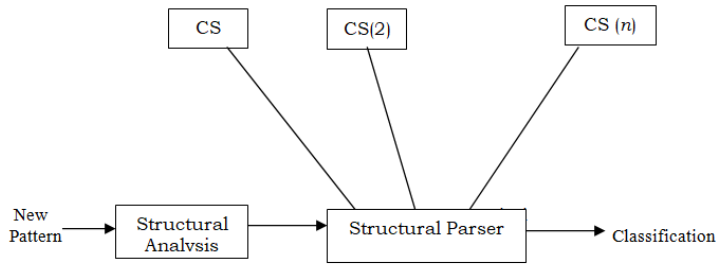
probabilistic in nature or not even approximately stochastic. Given the power of existing computing system it seems very appropriate and promising to find non-probabilistic non-dichotomous models and structures that enable us to recognize and transmit in a usable form patterns of this type, which humans cannot find without the help of more powerful methods than *eyeball-search*.

The promise of Syntactic Pattern Recognition is that the structure of the pattern, so far ignored, is now the most important element in the recognition process. This structure is used for two purposes:

(a) Describing the pattern

(b) Classifying the pattern.

A general scheme of a Syntactic Pattern Recognition system is shown in the Figure:



Classification using Syntactic Pattern Recognition

We consider n different classes and each of them is associated with a specific *structure*: Class Structure (CS), which is typical only to patterns in this class. Each unknown pattern is processed to obtain its structural analysis. Then starts the process of *structural parsing* in which the pattern structure is compared with the existing CS(1), CS(2),…,CS(n). If a match occurs with CS(*i*), the pattern is classified in class *i*. Otherwise it is rejected.

Most techniques in Syntactic Pattern Recognition are based on transforming complex patterns using hierarchical decomposition into simpler subpatterns, just as a sentence in a natural language may be decomposed into words (and then into letters). This process of decomposing may continue several times until we obtain the *pattern primitives*, which are not being decomposed further.

2. SYNTACTIC PATTERN RECOGNITION

Consider a two-class pattern problem. Let the patterns of these classes, C_1 and C_2 be composed of features from a set of terminals V_T. Thus, each pattern may be regarded as a sentence since it is composed of terminals. Let G be a grammar such that its language $L(G)$ consists of patterns (sentences) which belong to C_1. Then, any incoming pattern can be classified in C_1 if it belongs to $L(G)$, otherwise it will be classified as belonging to C_2.

Example 1

Consider a two CF grammer $G = \{V_T, V_N, P, S\}$ where $V_T = \{a, b\}$, $V_N = \{S, A\}$ and the production set is

$P : S \to aSb$

$S \to b$

The language $L(G)$ consists of the strings $\{b; a^n b^{n+1}, n \geq 1\}$. If a two-class classification problem is such that C_1 includes only the patterns $\{b; a^n b^{n+1}, n \geq 1\}$ which C_2 includes only the patterns $\{a^n b^{n+1}, n \geq 1\}$ we can classify an incoming pattern *x* using the following rule:

$x \in C_1$ *iff* $\mathrm{x} \in \mathrm{L}(\mathrm{G})$

$x \in C_2$ *iff* otherwise

The procedure, which has to answer the question, whether or not a giving string is grammatically correct, is called parsing.

Generally a deal with m classes $\{C_i\}_1^m$ and associated languages $\{L_i\}_1^m$ formed by grammars $\{G_i\}_1^m$. An incoming pattern *x* is decomposed and is classified in C_i if it is a sentence in L_i .

3.SELECTING PRIMITIVES

The selection of primitives, by which the patterns of interest are going to be described, depends on the type of data and the associated application. The important requirements are that the primitives provide reasonable description of the patterns with respect to their structural relations and that they can also be easily recognized by nonsyntactic methods, since their own structural formation is not important.

Example 2

Consider the problem of separating between all rectangles and all the other four sides polygons. We select the primitives.

a: 0^0 horizontal edge

b: 90^0 vertical edge

c: 180^0 horizontal edge

d: 270^0 vertical edge

and set of all rectangles will be represented by a string abcd. If we want to distinguish between rectangles of different sizes we select as primitives (edges) a_0, b_0, c_0, d_0 of length 1 pointing at the same directions. The set of all rectangles is

$$L = \{a_0^n \; b_0^m \; c_0^n \; d_0^m \; ; n, m = 1,2,3,..\}$$

Remark: Once set of primitives has been chosen for the pattern, we need to design a grammar whose language will describe the training patterns. As much as it would be desirable to obtain such a grammar automatically from the string of primitives, which describe the patterns, it is usually the user who constructs an appropriate grammar based on personnel knowledge and experience.

4. SYNTAX ANALYSIS FOR RECOGNITION

Once a grammar is designed we want to construct a pattern recognition system that will recognize the patterns generated by the grammar. As remarked earlier, a straightforward approach is to construct a specific grammar associated with the classes C_i ,1≤ i ≤ m respectively. Let *x* be an incoming unknown pattern given as a string. The recognition problem is finding $L(G_i)$ such that$x \in L(G_i)$.

The process of determining $L(G_i)$ is called syntax analysis or parsing. Apart from giving an answer to the classification problem, syntax analysis also provides the tree associated with x. The process itself can be described as follows:

Given a sentence *x* and a Grammar G, construct a triangle with top vertex S and bottom side *x* (as shown below) inside which we fill the derivatives tree.

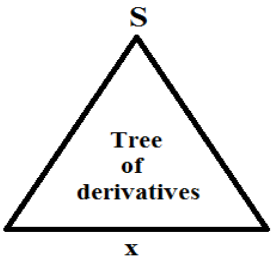


Figure : Parsing X

If we succeed and get *x* at the bottom of the tree then$x \in L(G_i)$. The tree can be found either by starting from the top S (top–down) parsing or by starting from the bottom x- (bottom-up) parsing.

The parsing process can be very slow if all the possible trees are considered. Very seldom we will be in a position where only choice is available at every step. Usually, we have several choices and must find a way to ignore those choices that will eventually lead to nowhere.

Top Down Parsing

Consider a grammar G with $V_T = \{a,b,c,d,e\}$, $V_N = \{S,A,B\}$ and productions

$S \to A$	$A \to BeA$	$B \to a$
$S \to Ads$	$A \to B$	$B \to b$
		$B \to c$

And let *x* = adbec a given sentence. To parse *x* top–down, we start from S and choose (the only correct choice as can be easily deduced) the production $S \to Ads$. At the next tree level we must choose $A \to B$ and $S \to A$ otherwise we end with sentences which are not x. The next step must be $B \to a$ and $A \to BeA$. This follows from $B \to b$ and $A \to B$. The finial step is $B \to C$. The different steps of the parsing are shown in the following figures:

TOP DOWN PARSING

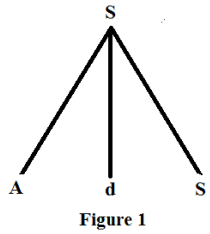


Figure 1

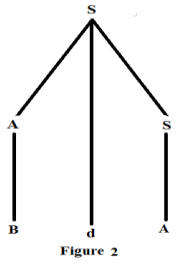


Figure 2

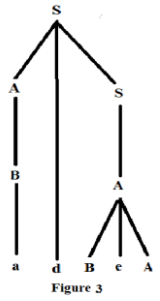


Figure 3

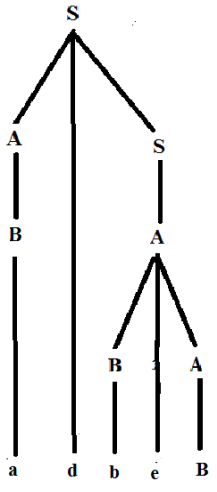


Figure 4

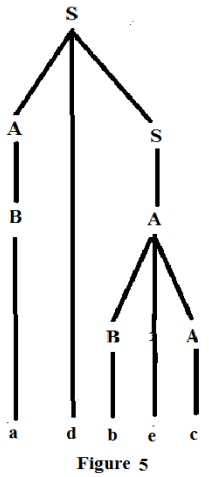


Figure 5

Bottom Up Parsing

Consider the same example used earlier to explain top down parsing. The following procedure outlines the basics of bottom up parsing. At each step we denote by S' final set of end nodes of the tree. A subset of S' which is a set of leaves (end nodes) of a subtree of the current derivation tree is called a phrase. The left most is called the handleof S'. From figure (5) we obtain the sentence adbec with phrases a, b, c, bec, adbec. The handle here is a. The bottom-up parsing process states with the final sentence S_0 and repeats the following steps:

(1) Find the handle of S_i

(2) Delete the handle, subject to the production set and obtain S_{i+1}

In the example,

S_0 =adbec S_1 =Bdbec S_2 =Adbec S_3 =AdBec

S_4 =AdBeB S_5 =AdBeA S_6 =AdA S_7 =AdS

S_8 =S

In bottom up left to right parsing, there can be at each step many strings that can be replaced by non-terminals, thus forcing us to try many possibilities. Some grammars called $LR(k)$ [knuth (1965)] garmmers enable the parsing process to be deterministic, provided that it is always possible to look *k* symbols beyond the current one.

5. STOCHASTIC LANGUAGES

Due to measurement noise and some ambiguity regarding the characteristics of the pattern classes, it is necessary to consider a stochastic model of grammar and stochastic languages.

Definition: A stochastic grammar is a set $G_s = \{V_T, V_N, P, Q, S\}$ where V_T, V_N, P and S are as explained in preliminaries, and Q is a set of probabilities associated with the given production.

Example: Let $V_T = \{a, b\}, V_N = \{S\}$ and $(P,Q) = \begin{cases} S \xrightarrow{p} aSb \\ S \xrightarrow{1-p} ab \end{cases}$

By applying the first and second productions alternatively we obtain a sentence $x = a^2b^2$ whose probability is $p(x) = 1 - p$. If *x* can be obtained in several ways, its probability is adjusted accordingly.

Definition: A stochastic language $L_s(G_s)$ is a language generated by a stochastic grammar.

In order for the set Q to be consistent we must have $\sum_{x \in L_s} p(x) = 1$.

Assume that $x \in L_s$ is generated from S by $S \xrightarrow{p_1} \alpha_1 \xrightarrow{p_2} \alpha_2 \ldots\ldots\ldots \xrightarrow{p_n} \alpha_n = x$

where $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$ are intermediate strings and α_{i+1} is obtained from α_i using a production rule P_{i+1} with an associated probability $p_i = p(p_i)$. Then the probability to obtain *x* is

$$p(x) = p(p_1)\, p(p_2/p_1) \ldots p(p_n/p_1, p_2, \ldots p_{n-1})$$

It always $p(p_n / p_1, p_2, \ldots p_{n-1}) = p(p_i)$, the production p_i is called unrestricted. The knowledge of the production probabilities reduces the time consuming of the parsing process for stochastic languages.

REFERENCES

Bezdek (1981), Pattern Recognition with Fuzzy Objective Function Algorithms Springer, New York

Verhagen.C (1975) Pattern recognition, Volume 7, Issue 3, September 1975, Pages 109-116 Elsevier

Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. Academic Press, 2nd edition, New York

Schalkoff, R. J. (1992). *Pattern Recognotion: Statistical, Structural and Neural Approache*s. Wiley, Singapore.

MineichiKudo and Jack Sklansky (2000).Comparison of algorithms that select features for pattern classifiers. Pattern Recognition, Vol. 33 (1) pp. 25-41.

Johnson Richard A., and Wichern Dean.W(2001). Applied Multivariate Statistical Analysis, Prentice Hall International Inc., Englewood Cliffs, New Jersey.

A.V.DattatreyaRao, and N.SrinivasaRao (2005) Linear Classifiers in Statistical Methods, Vol.7 No:1, pp.29-45.