SYNTACTIC PATTERN RECOGNITION

Dr. K.Vijayarekha
Associate Dean
School of Electrical and Electronics Engineering
SASTRA University, Thanjavur-613 401
Table of Contents

1. SYNTACTIC PATTERN RECOGNITION.......................... 3
1. SYNTACTIC PATTERN RECOGNITION

As we have seen in the previous lectures, a pattern recognition system design will have the following components

1. A feature vector $x$ is found
2. The system is trained using a training pattern whose classification is known before
3. Unknown patterns are classified using Crisp or Fuzzy Pattern Recognition approach.

In the real world scenario, the pattern contains some relational information from which appropriate feature vectors cannot be derived easily. The inter-relationships between patterns are ignored to some extent and hence the pattern recognition system may fail.

The structure of the pattern recognition system assumes extreme significance and has two purposes

1. Description of the pattern
2. Classifying the pattern
Most syntactic pattern recognition systems are based on decomposition of big hierarchical patterns into smaller sub-patterns. For example, consider sentence decomposition into words and then letters as an analogy. This process of decomposition may continue several times till we obtain pattern primitives which cannot be decomposed.

N classes are considered and each of them is associated with a specific structure, Class Structure (CS) which is typical to each pattern in the class. Each pattern is processed to obtain its structural analysis. Then the process of structural parsing begins in which the existing pattern is compared to the existing one CS(1), ………. CS (n). If a match occurs for CS (i), then the pattern is classified as i, else it is rejected.

A pattern’s structure is described by a specific language. The collection of rules which determine conversion of primitives to patterns is called Grammar, specific to the language.

Now, let us see some preliminary definitions with respect to syntactic pattern recognition systems.

1. An Alphabet is a finite set of symbols

   \[ V = \{ x_1, x_2, \ldots, x_n \} \]

2. A sentence over V is a finite set of ordered symbols (from left to right) from V

3. For arbitrary strings \( s_1 = x_1x_2\ldots x_m \) and \( s_2 = y_1y_2y_3\ldots y_n \), the concatenation of strings is given by

   \[ s_1 \circ s_2 = s_3 = x_1x_2\ldots x_m y_1y_2y_3\ldots y_n \]

   where length of \( s_3 \) is \( m+n \)

4. An arbitrary set of \( V^* \) is \( L \) which is called language

5. For a set of arbitrary languages \( L_1 \) and \( L_2 \), the set

   \[ L_1 \circ L_2 = \{ s \mid s = s_1s_2, s_1 \in L_1 \text{ and } s_2 \in L_2 \} \]

   is called the concatenation of \( L_1 \) and \( L_2 \)

6. For an arbitrary set \( s, t \in V^* \), \( s \) is called the substring of \( t \) if
For strings $u, v \in V^*$

Having seen the preliminaries to an extent, let us see in more detail, the concept of GRAMMAR

A Grammar is defined as a four entity substance

$$G = \{V_T, V_N, P, S\}$$

Consisting of

1. A set of terminal symbols called primitives, which are denoted by $V_T$ which is a subset of alphabet $V$
2. A set of non-terminal variables which are used as intermediate quantities in the process of getting an arbitrary outcome. This outcome is composed of terminal symbols i.e., of constants, contrary to the non-terminal symbols called variables.
3. A set of production rules, often called production or rewriting rules. This set is coupled with $P$ denoted by $V_T$ provides the structure of the given grammar.
4. A starting symbol denoted by $S$ where $S \in V_N$

TYPES OF GRAMMAR:

1. Unrestricted Grammar (UR) has no restrictions on its productions. Each Production is given by $\alpha \in V^+$ and $\mu \in V^*$.
2. A context sensitive Grammar (CS) allows only productions of the form
   $$\alpha A_1 \mu \rightarrow \alpha \mu_2 \mu$$
   Where $\alpha, \mu \in V^*$, $A_1 \in V_N$, $\mu_2 = V^* - \epsilon$
3. Context Free Grammar (CF) has productions of the form
   $$A_1 \rightarrow \mu_2$$
4. A finite state Grammar has the form
   $$A_1 \rightarrow a \text{ or } A_1 \rightarrow aB_2$$
   Where $A_1, B_2 \in V_N$ and $a \in V_T$

Let us now use these definitions to form a syntactic pattern recognition problem:
Consider a 2 class pattern recognition problem, namely $C_1, C_2$ but composed of features from a set $V_T$. We can regard each pattern as a sentence since it composed of terminals. Let $G$ be the grammar such that its language $L(G)$ is composed of elements consists of patterns (sentences) which belong to $C_1$. Then by that extension any incoming pattern can be classified in $C_1$ if it belongs in $L(G)$. Otherwise it will be classified as $C_2$.

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

\[
P: S \rightarrow asb
\]

\[
S \rightarrow b
\]

The language $L(G)$ consists of the strings $\{b; a^n b^{n+1}, n>1\}$. If a 2 class problem is such that $C_1$ includes only the patterns while $C_2$ includes only the patterns $\{a^n b^n, n>1\}$, we can classify an incoming pattern $x$ based on the following

\[
X \in C_1 \text{ iff } x \in L(G)
\]

\[
X \in C_2 \text{ otherwise}
\]

The procedure which has to answer whether the string is grammatically correct or not is called parsing