DISCRIMINANT GRAMMARS FOR PATTERN CLASSIFICATION

Dissertation for the Degree of Ph. D.

MICHIGAN STATE UNIVERSITY
ALAN JAMES FILIPSKI

1976

This is to certify that the

thesis entitled

Discriminant Grammars for Pattern Classification

presented by

Alan James Filipski

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in **Computer Science**

Carl V. Page
Major professor

Date **May 17, 1976**

O-7639

ABSTRACT

DISCRIMINANT GRAMMARS FOR PATTERN CLASSIFICATION

By

Alan James Filipski

This thesis introduces the idea of a "discriminant grammar" as a tool for syntactic pattern recognition. A discriminant grammar is defined as a context-free, unambiguous grammar together with a mapping from the productions into the reals. We associate a number with each sentence generated by the grammar by adding the numbers corresponding to each use of a production in the derivation of the sentence. The language is partitioned into three decision regions by comparing this sum to a cutpoint. It is shown that a discriminant grammar may be used to provide a sufficient statistic for decision between two stochastic grammars. Results are given in terms of a Bayesian formulation and a sequential scheme using top-down parsing and LL(k) grammars. It is shown that regular discriminant grammars can yield decision regions that are not recursively enumerable, but if all coefficients are rational, the regions are context-free. Results are obtained concerning the relationship of regular discriminant grammars to probabilistic automata. The use of perceptron-like methods to learn the coefficients from empirical samples is also discussed.

DISCRIMINANT GRAMMARS
FOR
PATTERN CLASSIFICATION

By

Alan James Filipski

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1976

ACKNOWLEDGEMENTS

I would like to express thanks to all members of my committee, especially Dr. Richard Dubes for his careful reading of the various drafts of this thesis and my chairman, Dr. Carl Page, for his encouragement and assistance during the many false starts and doldrums preceding the idea which led to this work.

I would also like to express thanks to those who helped in the typing of this thesis, namely Bob Frye and Annette Davis.

Special thanks are due to my wife, Lois, for her encouragement throughout my long period of graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The term "Pattern Recognition" as applied to the more classical feature-space oriented techniques as well as to the newer syntactic methods, has tended to obscure a fundamental difference of intent between the two approaches. The former is almost exclusively concerned with the development of algorithms to extract or utilize discriminatory information, that is, information which pertains to the assignment of an object to one of several classes. Any information which does not further this end is considered a nuisance. Most work in "Syntactic Pattern Recognition", on the other had, does not stress this distinction. The "Syntactic Pattern Recognizer" is generally expected to transduce the input object into a derivation, which is a representation of all structural information inherent in the original object in terms of some given grammar. This derivation may then be interrogated to answer questions about the object, produce a description of the object, etc. We can thus distinguish between "Pattern Classification" and "Pattern Description", the former being primarily a process of information reduction and the latter a process of information re-organization.

Both of these are useful ends toward which syntactic means may be applied. They are, however, distinct ends and

efficiency may be sacrificed if we confuse them. As an extreme example, suppose we are given a character string and we must decide whether it is a Fortran program or an Algol program. A very inefficient way to make this decision would be to feed the string into a Fortran compiler and then into an Algol compiler and then note which generated fewer error messages. It is obvious that the decision could be made instead by a very small and fast program which looked for a few characteristic structures. The important assumption here, of course, is that we are interested only in a decision and not in any by-products such as error messages or object programs.

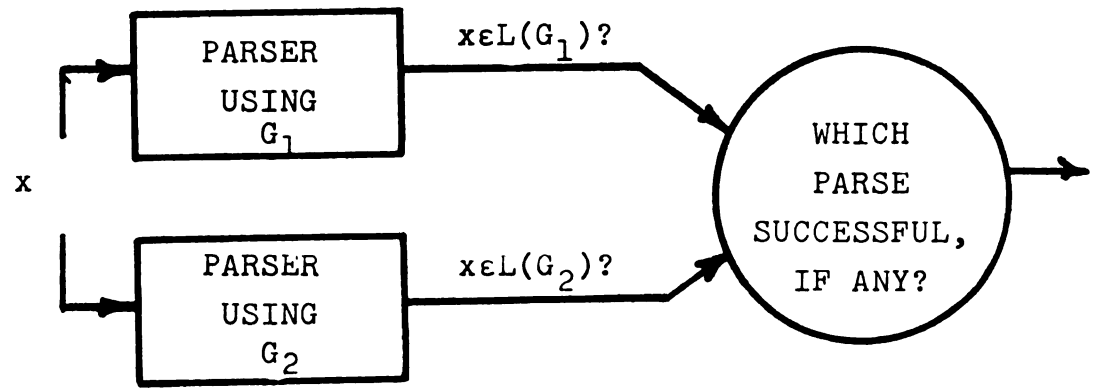
It is the object of this work to show that syntactic methods may be applied to a purely classificatory end, i.e. that there is such a thing as structural discriminatory information and that this information may be effectively extracted and used by syntactic means. Hopefully, restricting our attention ty discriminatory information will serve to make the path to the decision itself a direct one.

It is evident that in many applications for which the use of syntactic methods has been proposed, only the ultimate decision is of any importance, and additional information supplied by the parse represents wasted effort. Examples are hand-printed character identification and automatic blood-cell counting. Most systems used on a production basis need only supply a decision (or possibly

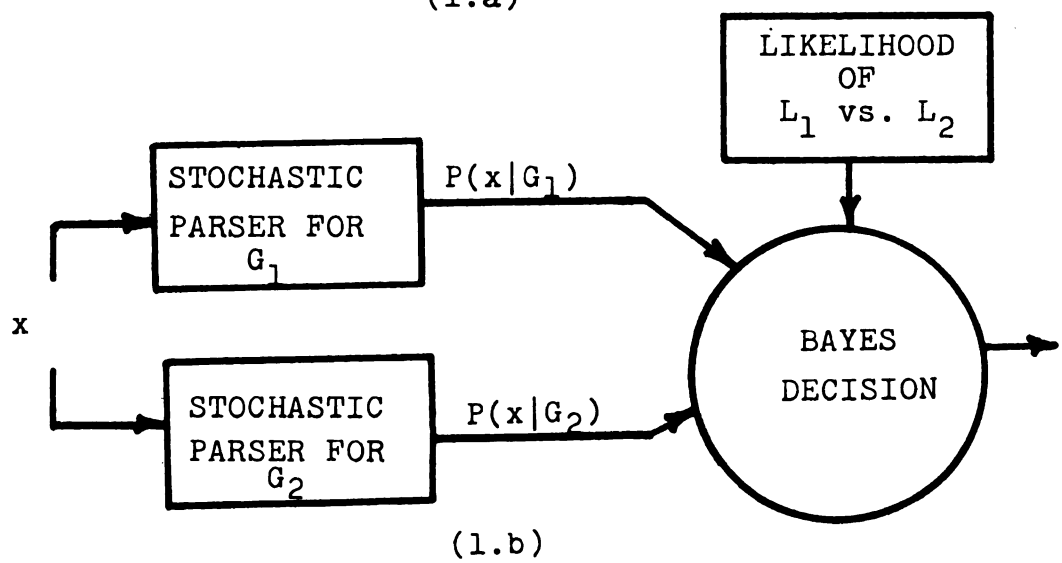
report inability to make a decision). See, for example, Kanal(12).

Another instance in which descriptive information in addition to a decision is of very little value occurs when the grammar is obtained by means of the semi-automatic inductive methods now being explored by a number of researchers. (See Fu & Booth(8) for a survey.) In this case there would be no a priori correspondence of semantic with syntactic notions with the result that knowledge of the derivation becomes rather useless. In this case we should simply ask the system to report a decision rather than to exhibit a parse in an unfamiliar grammar.

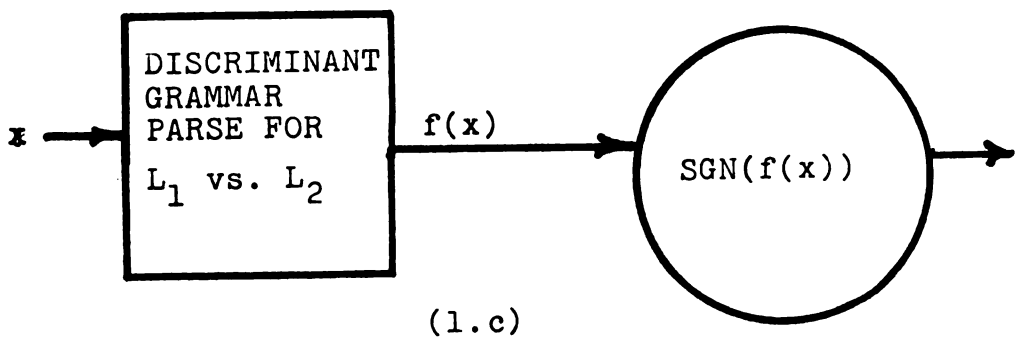
Figure 1 depicts three possible approaches to syntactic pattern classification. Figure 1.a represents the most straightforward approach: The string is processed through a parser for each grammar; one, none, or both of the parsers then report success. Figure 1.b is a modification of this in the case where probabilistic information is available. In this case two stochastic parses yield the probabilities that the string was generated by each of the two stochastic grammars. This information, together with the a priori likelihoods of the two classes, is fed into a Bayes decion-maker. These two methods are used, for example, by Lee & Fu(15). Figure 1.c represents the approach introduced in this thesis. A single parse is performed using a "discriminant grammar" which maps the string x into a single real number, say f(x). The sign



(1.a)



(1.b)



(1.c)

Figure 1 Syntactic Decision Strategies

of this number then determines the classification of x. The absolute value of f(x) is a measure of our confidence in the classification. (Under a statistical interpretation of the discriminant grammar to be given later, f(x) relates to a quantity which I. J. Good calls the "amount of information in an observation about a hypothesis". (Good(9), Kullback(14)))

The potential advantages of the discriminant grammar approach are several:

1.) Only a single parse need to made. In fact, as we shall see later, if the user wishes to specify error tolerances, this parse need not even be completed.

2.) A discriminant grammar can usually be made simpler than a grammar describing either of the languages because only discriminatory information need be considered. For example, the two languages

$$L_1 = \{a^i b^i | i \geq 1\}$$

$$L_2 = \{b^i a^i | i \geq 1\}$$

are both context-free, but a discriminant grammar with <u>regular</u> underlying grammatical structure can be used to discriminate between them on the basis of whether the first character of a given string is an a or a b.

3.) The set of strings for which the discriminant grammar will yield a decision can be made larger than the union of

the two original languages, thus giving a means for the classification of noisy strings. As a trivial example, suppose we want to distinguish between the languages {a}* and {b}*. It will prove to be a simple matter to produce a discriminant grammar (again, with regular underlying structure) which will be able to classify any string from {a,b}* depending upon whether there are more a's or more b's in the string.

The discriminant grammar approach may be regarded as a step towards a synthesis of the syntactic and feature-space formulations of the Pattern Recognition problem. We will show later how syntactic information may be represented by means of a mapping from strings into a space of structural indices. Once this transformation is made, we then may apply results from feature-space theory, as presented, for example, in Duda & Hart(4). It is hoped that this "structural statistics" approach will help the two methodologies of Pattern Recognition to cross-fertilize each other.

One of the most important areas of syntactic Pattern Recognition concerns the processing of two-dimensional pictures. Since the early 1960's (e.g. Minsky(18)) there has been considerable work done on languages for the representation of various types of two-dimensional images. Freeman(5) proposed a language of concatenated vectors to describe connected curves. Kirsch(13) suggested a grammar of two-dimensional arrays. Web grammars, Plex grammars,

Tree grammars and a number of picture description languages have been proposed (see Fu(7) for a survey) in addition to many special purpose languages, e.g., for structural formulae, flowcharts, and mathematical expressions.

In some cases these grammars transduce the input picture into a one-dimensional string which can then be dealt with using all the apparatus of formal language theory. Methods of this type have been quite successful (e.g. Lee & Fu(15)). Other grammars, such as the Web and Plex grammars, seem to demand a more general notion of parsing. This thesis follows the lead of the former approach and is set in the classical theory of formal languages. It is easy to conceive of situations in which two-dimensional syntactic classification schemes may be put to use. In the game of Go, for example, a board configuration is essentially syntactic in that it is the spatial relationships between primitives (stones) which is of significance in determining the worth of a given board position to either player. Furthermore, if our task were to develop a static evaluation function for the game we would be interested only in the relative value of a position and not in its full syntactic description. This is an example of the possible use of a two-dimensional extension of the discriminant grammar technique.

II. DISCRIMINANT GRAMMARS-DEFINITION AND INTERPRETATIONS

A. DEFINITION

The basic structure introduced and developed in this thesis for use in syntactic pattern recognition is the "discriminant grammar" or "d-grammar". Formally, a discriminant grammar D is defined to be an ordered quintuple

$$D = (V_N, V_T, \Pi, S, R)$$

The first four components are, respectively, a set of non-terminal symbols, a set of terminal symbols, a set of production rules, and a start symbol. It is assumed that the reader is familiar with the concepts and notational conventions of formal language theory as contained in, for example, Hopcroft and Ullman(10). The component R is defined as a mapping from the production set Π into the real numbers. It is often convenient to use the notation r_i for $R(\pi_i)$, where $\pi_i \epsilon \Pi$. The ordinary phrase-structure grammar defined by the first four components of D will be denoted CHAR(D) and will be called the characteristic grammar of D.

Some restrictions are usually put on the form of the production rules of a grammar. The restriction that has been found to strike a good balance between generative

power and mathematical tractability is the context-free restriction, i.e. that the premise of each production rule consist of a single non-terminal symbol. In addition, it is reasonable to require that each string in the language have an essentially unique derivation in the given grammar. This property is called unambiguity. We will limit the grammars under consideration as follows: Unless otherwise noted, all discriminant grammars used in this thesis will be assumed to have unambiguous, context-free characteristic grammars.

A mapping Q from L(CHAR(D)) into the reals is defined by

$$Q_D(x) = \sum_{k=1}^{n} r_{i_k}$$

where $\pi_{i_1}, \pi_{i_2}, \pi_{i_3}, \cdots \pi_{i_n}$ is the unique left-most canonical derivation (LMCD) of the string x. (Throughout this thesis, when we speak of "derivations" we shall mean sequences of productions, not sequences of sentential forms.) Note that it does not in fact matter whether we use the LMCD to compute Q(x) or whether we use any derivation of x, since all derivations of x in the unambiguous characteristic grammar are simply permutations of each other. When considered as set of ordered pairs, the function Q is called the discriminant language generated by the discriminant grammar D and is denoted L(D), i.e.

$$L(D) = \{ (x,Q(x)) \mid x \in L(CHAR(D)) \}$$

Given any real number θ (a "cutpoint"), the language L(CHAR(D)) may be partitioned into three classes in the following way:

$$L^{+}(D,\theta) = \{x \mid x \varepsilon L(CHAR(D)) \text{ and } Q(x) > \theta\}$$

$$L^{0}(D,\theta) = \{x \mid x \varepsilon L(CHAR(D)) \text{ and } Q(x) = \theta\}$$

$$L^{-}(D,\theta) = \{x \mid x \varepsilon L(CHAR(D)) \text{ and } Q(x) < \theta\}$$

This partition will be interpreted as a simple decision scheme which will allow us to distinguish between two languages. It is sometimes conceptually convenient to decompose this decision scheme into four stages: structural indexing, projection, linear translation and quantization. It is possible in this way to express the decision as a composition of four functions:

$$x \varepsilon L^{i}(D,\theta) \text{ iff } Sgn(T(P(S(x)))) = i$$

Where S, P and T are described as follows: Let N be the set of non-negative integers. For any discriminant grammar $D = (V_N, V_T, \Pi, S, R)$, let k be the number of productions in Π, i.e.

$$\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$$

For any $x \varepsilon L(CHAR(D))$ and for i = 1 to k, let m_i equal the number of times π_i occurs in the LMCD of x.

Then define $S : L(CHAR(D)) \rightarrow N^k$ as $S(x) = (m_1, m_2, m_3, \ldots, m_k)$.

We may interpret S as a mapping from L(CHAR(D)) into a space of structural indices determined by the underlying characteristic grammar. Note that even though CHAR(D) is unambiguous, the mapping S is not necessarily one-to-one, as the following example shows:

EXAMPLE Let CHAR(D) be $G = (\{S,A\},\{b,c\},\Pi,S)$

where Π consists of the productions

$S \rightarrow AA$

$A \rightarrow b$

$A \rightarrow c$

Then $S(bc) = S(cb) = (1,1,1)$

The projection function $P : N^k \rightarrow (-\infty, +\infty)$ is defined as

$$P(m_1,m_2,\ldots,m_k) = \sum_{i=1}^{k} m_i r_i$$

Finally the translation function $T : (-\infty, +\infty) \rightarrow (-\infty, +\infty)$ is defined as

$$T(\xi) = \xi - \theta$$

(Note that T is actually a function of θ also.) The output of this function is then quantized at zero yielding the decision.

This exposition of simple discriminant grammar decision-making in terms of a composition of many-to-one

functions is valuable because it defines exactly what sort of information-reduction is performed at each stage of the process from observation to decision. Clearly the most interesting function of this chain, and the one which is unique to this thesis, is the structural indexing function S.

The functions S and P are implicit in the specifications of the discriminant grammar, while the function T depends upon some specified cutpoint.

As a simple example of a discriminant grammar, consider the following:

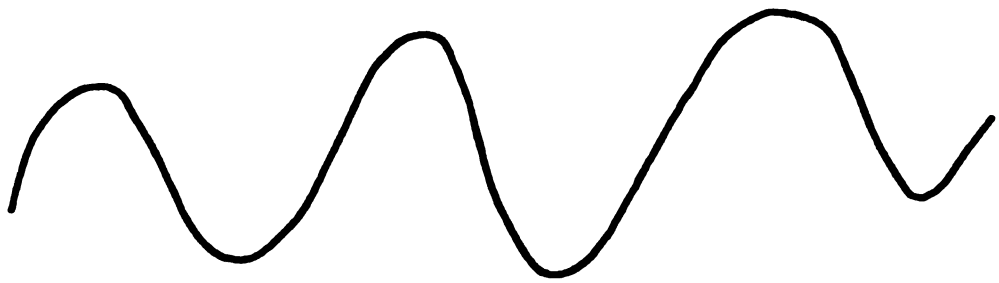
EXAMPLE Let $D = (\{S,A,B\},\{a,b\},\Pi,R)$

Where Π and R are given by the following table:

π_i	$R(\pi_i)$
$S \rightarrow aA$	0
$S \rightarrow bB$	0
$S \rightarrow \varepsilon$	0
$A \rightarrow aA$	+1
$A \rightarrow bB$	-1
$A \rightarrow \varepsilon$	0
$B \rightarrow aA$	-1
$B \rightarrow bB$	+1
$B \rightarrow \varepsilon$	0

Given any string $x\varepsilon\{a,b\}^*$, $Q_D(x)$ tends to be positive if x contains long "runs" of either a's or b's and tends

to be negative if the a's and b's tend to alternate with a short period. A concrete interpretation of this discriminant grammar would be to consider the a's and b's respectively as positive and non-positive slopes between appropriately sampled points of some periodic function over some interval. Then for some arbitrary cutpoint θ, $L^{+}(D,\theta)$ would consist of relatively low frequency functions while $L^{-}(D,\theta)$ would contain functions with predominantly higher frequency components. Figure 2 gives an example of such a situation. The functions are sampled at the "0" point of the axis and at each "+" point. The sampled value at each "+" point is compared to the previously sampled value. If there has been an increase, an a is inserted into the string, otherwise a b is inserted. For example, the first character of 2.a is an a since the ordinate of the curve is greater at the first "+" than at the origin. The encodings and their corresponding Q-values are given in the figure. Thus we see that the string represented in Figure 2.a is contained in $L^{-}(D,0)$ while the string represented in Figure 2.b is contained in $L^{+}(D,0)$.



0....+....+....+....+....+....+....+....+....+....+....+

2.a x = abaabbaaba $Q_D(x) = -3$

0....+....+....+....+....+....+....+....+....+....+....+

2.b x = aaabbbbbbb $Q_D(x) = 7$

Figure 2 Encoded Line Patterns

B. A BAYESIAN INTERPRETATION

In this section we will investigate the use of the discriminant grammar in making a Bayes decision between two stochastic languages. (See the Appendix for definitions and notations involving stochastic grammars.)

Definition Two stochastic grammars G_1 and G_2 are said to be commensurate if and only if $CHAR(G_1) = CHAR(G_2)$ and all production probabilities are nonzero under both G_1 and G_2.

Definition A discriminant grammar D is said to be a log-likelihood ratio representation of two commensurate stochastic grammars G_1 and G_2 if and only if:

1. The characteristic grammars of G_1, G_2 and D are the same.
2. For all $\pi_i \varepsilon \Pi$,
   $R(\pi_i) = \log P_1(\pi_i) - \log P_2(\pi_i)$
   Where P_1, P_2 are the probability functions of G_1 and G_2 respectively.

If D is the log-likelihood ratio representation of G_1 and G_2, we write $D = LLRR(G_1, G_2)$. Note that in general, $LLRR(G_1, G_2)$ does not equal $LLRR(G_2, G_1)$.

Suppose now that we are given a string x and two commensurate stochastic grammars G_1 and G_2 such that x can

be generated by their common characteristic grammar. Let $\Gamma_i(x)$ be the probability of generating x under G_i. Let the a priori probabilities of G_1 and G_2 be denoted $Pr(G_1)$ and $Pr(G_2)$. Let H_i be the hypothesis that x was generated under G_i. Finally, let L_{ij} be the loss incurred in deciding H_j when the true hypothesis actually is H_i.

Given any x, the conditional expected loss incurred in deciding H_1 is given by:

$$t_1(x) = L_{21}Pr(G_2\ x) + L_{11}Pr(G_1\ x)$$

Using Bayes rule, by which

$$Pr(G_i|x) = \frac{Pr(G_i)\ \Gamma_i(x)}{Pr(x)}$$

We obtain

$$t_1(x) = \frac{L_{21}Pr(G_2)\ \Gamma_2(x)+L_{11}Pr(G_1)\ \Gamma_1(x)}{Pr(x)}$$

Similarly, the conditional expected loss incurred in deciding H_2 is given by

$$t_2(x) = \frac{L_{12}Pr(G_1)\ \Gamma_1(x)+L_{22}Pr(G_2)\ \Gamma_2(x)}{Pr(x)}$$

We can minimize the expected loss by deciding

1) H_1 if $t_1(x) < t_2(x)$

2) H_2 if $t_2(x) > t_1(x)$

3) making a random decision if

$$t_1(x) = t_2(x)$$

This is the same as deciding H_1 if

$$(L_{21}-L_{22})Pr(G_2)\ \Gamma_2(x) \ < \ (L_{12}-L_{11})Pr(G_1)\ \Gamma_1(x)$$

If we assume that $L_{ij} > L_{ii}$ when $i \neq j$, this inequality reduces to

$$\frac{\Gamma_1(x)}{\Gamma_2(x)} > \frac{Pr(G_2)}{Pr(G_1)} \cdot \frac{(L_{21}-L_{22})}{(L_{12}-L_{11})}$$

Taking logs of both sides and expanding Γ_i yields the rule:

decide H_1 if

$$\sum_{k=1}^{m} (\log P_1(\pi_{i_k}) - \log P_2(\pi_{i_k})) > \log \frac{Pr(G_2)\ (L_{21}-L_{22})}{Pr(G_1)\ (L_{12}-L_{11})}$$

where $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_m}$ is the LMCD of x in $CHAR(G_1)$.

Now let $D = LLRR(G_1,G_2)$ and let

$$\theta = \log \frac{\Pr(G_2)\ (L_{21}-L_{22})}{\Pr(G_1)\ (L_{12}-L_{11})}$$

Then we have shown that the unconditional expected loss (Bayes risk) is minimized by the rule:

if $x \varepsilon L^{+}(D,\theta)$ then decide H_1

if $x \varepsilon L^{-}(D,\theta)$ then decide H_2

if $x \varepsilon L^{0}(D,\theta)$ then decide arbitrarily.

Using a loss function of $L_{12} = L_{21} = 1$ and $L_{11} = L_{22} = 0$, the Bayes rule minimizes the probability of misclassification. This Bayes error may be expressed as

$$P(\text{err}) = \sum_{x \varepsilon L(\text{CHAR}(G_1))} \min\ [\Pr(G_1)\ \Gamma_1(x),\ \Pr(G_2)\ \Gamma_2(x)]$$

The size of this probability is in general quite difficult to compute. In special cases, however, we may obtain bounds on the value of P(err). One such case is discussed in the next section.

C. BOUNDS ON THE BAYES ERROR (LINEAR GRAMMARS)

In this section we present a technique for the calculation of upper and lower bounds on the Bayes error (P(err)) for linear grammars using the Bhattacharyya coefficient.

Definition A context-free grammar is linear if an only if the consequence of each production (i.e. the right hand side of that production) contains at most one non-terminal. Thus each production must be of the form

$$A \rightarrow xBy$$

or

$$A \rightarrow x$$

where $x \epsilon V_T^*$, $y \epsilon V_T^*$

Definition Given two probability mass functions p(.) and q(.) defined on the same discrete sample space X, the Bhattacharyya coefficient ρ of p vs. q is

$$\rho = \sum_{x \epsilon X} \sqrt{p(x)q(x)}$$

It is known (see Kadota & Shepp(11)) that the Bhattacharyya coefficient can be used to form an upper and

lower bound on P(err) as follows:

$$\rho^2 \min(Pr(G_1), Pr(G_2))/2 \leq P(err) \leq \rho/2$$

We now solve the following problem: Given two commensurate stochastic grammars $G_1 = (V_N, V_T, \Pi, S, P_1)$ and $G_2 = (V_N, V_T, \Pi, S, P_2)$ with linear characteristic grammars, compute the Bhattacharyya coefficient of Γ_1 vs. Γ_2.

First, to each production $\pi_i \in \Pi$, assign a number

$$q_i = \sqrt{P_1(\pi_i) P_2(\pi_i)}$$

Then ρ may be expressed as

$$\rho = \sum \prod_{i=1}^{k} q_i^{m_i}$$

where the summation extends over all $x \in L(CHAR(G_1))$. As before, the m_i are the structural indices of x in the k-element production set. The key to the computation of this sum of products is the realization that the set of derivations of sentences in a linear grammar is itself a regular language over the production set Π. This fact will become apparent by the following construction. (For simplicity, we omit commas in the derivations.)

Suppose $V_N = \{A_1 (=S), A_1, A_2, \ldots A_m\}$ is the set of non-terminals for the linear grammar. Then the non-deterministic finite-state acceptor for derivations has m+1 states $\{s_1, s_2, \ldots s_{m+1}\}$ and on input π_i has a transition from s_j to s_k where A_j is the non-terminal in the premise of π_i and A_k is the non-terminal in the consequence of π_i; if the

consequence of π_i contains no non-terminals, then the transition should go to s_{m+1}. Let s_1 be the start state and s_{m+1} be the final state. Then a string $\pi_{i_1} \pi_{i_2} \cdots \pi_{i_r}$ is accepted by this automation if and only if it is a valid derivation of some string in the original linear grammar.

Note that the terminals in the original grammar have no bearing on the construction of this acceptor.

EXAMPLE Suppose G = ({A,B,C}, {d,e,f,g}, Π, A) where the productions are given by

i	π_i
1	A → eeB
2	A → dAgg
3	A → fb
4	B → g
5	B → eB
6	B → ggC
7	C → Afg
8	C → d

Then the corresponding acceptor for production sequences is given by Figure 3.

We now return to the original problem of calculating the Bhattacharyya coefficient. In the graph of the automaton just constructed, associate with each transition π_i the number q_i defined previously. Our problem is now that

of summing the products of the q_i along all possible paths from the start state to the accept state. To accomplish this, construct the (m+1) by (m+1) matrix W such that w_{ij} equals the sum of the q_k corresponding to all single-step transitions between s_i and s_j. In the example, for instance, W_{12} would be set equal to q_1+q_3. In addition to this, set $W_{m+1,m+1}$ equal to one. Now let W' be equal to the limit of W^n as n approaches infinity, if this limit exists. It is now an easily obtained combinatorial fact. (See, for example, Feller(3)) that $\rho = W'_{1,m+1}$

Note that this same technique may be extended to the class of grammars for which the number of non-terminals in every possible sentential form has some upper bound depending only upon the grammar. (This is the class of "ultralinear" grammars.) In this case, we simply assign a name to each group of non-terminals which can appear in a sentential form and follow the above procedure using this "super-non-terminal".

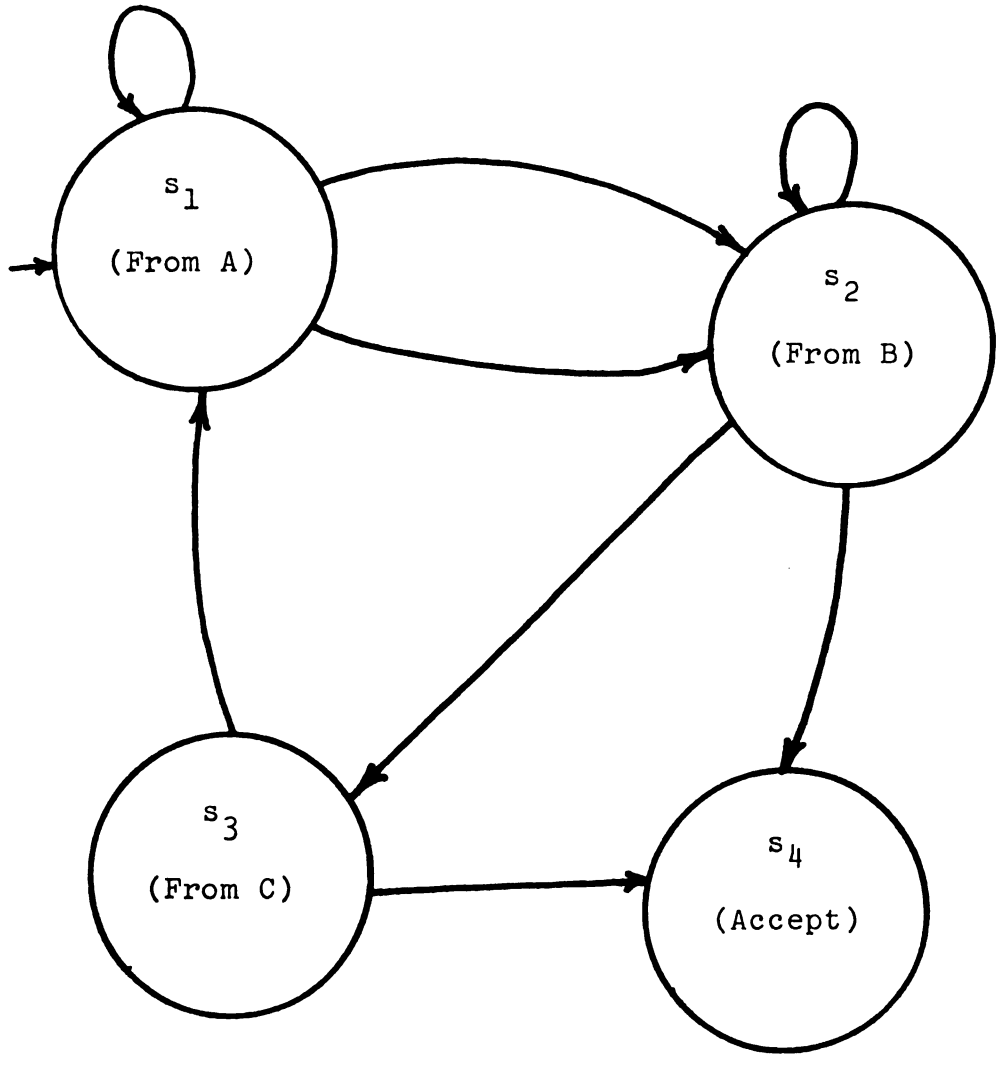


Figure 3 Acceptor for Production-Sequences

D. A SEQUENTIAL APPROACH TO DISCRIMINATIVE PARSING

D. 1. Introduction

Since Wald's introduction of the Sequential Probability Ratio Test (SPRT) (Wald(20)), the principle of optional stopping has become well known in the pattern recognition literature. See for example, Fu(6). The two principal approaches are the sequential Bayes approach and the SPRT. If only a finite number of features are available, these are not necessarily equivalent. The Bayes approach assigns costs to both feature measurements and incorrect decisions and then selects a scheme which minimizes the expected total cost. This is usually done computationally by backwards dynamic programming. The SPRT, on the other hand, is more closely related to Neyman-Pearson theory and stops observing features when the value of the likelihood ratio guarantees certain bounds on misclassification probabilities.

In this section we exhibit a scheme for sequential syntactic decision-making using the SPRT, discriminant grammars and LL(k) top-down parsing techniques. This scheme could save both parsing time, and, in the case of on-line systems, feature extraction time (and cost). The availability of such a sequential scheme is a unique

aspect of the discriminant grammar approach.

Informally, the essence of the scheme is this: We are given some string x and we want to decide which of two commensurate stochastic grammars generated x. We use the LL(k) parser to supply us (in top-down sequence and without backup) with the sequence of productions which forms the LMCD of the string x. Each time we are informed of a production in the derivation we decide either to request the next production from the parser or to abort the parse and make a decision immediately as to which grammar generated x. The stopping criteria will depend upon preassigned error tolerance.

We now proceed to lay the statistical foundations of the sequential parsing technique in some detail. Crucial issues in the development include the necessity for top-down parsing and the treatment of parses which terminate before reaching a stopping boundary.

Suppose that we are given two commensurate stochastic grammars $G_1 = (V_N, V_T, \Pi, S, P_1)$ and $G_2 = (V_N, V_T, \Pi, S, P_2)$. Denote their common characteristic grammar by G and let $D = \mathrm{LLRR}(G_1, G_2)$. For any string $x \in L(G)$, denote by H_i $(i = 1,2)$ the hypothesis that x was generated by G_i. As before, let $\Gamma_i(x)$ denote the probability of the generation of x under H_i.

Since derivations in G are strings over Π, we shall use the customary notations Π^* and Π^+ to mean, respectively, the set of all production sequences and the set of

all non-empty production sequences. The symbols σ, ρ and ω will usually be used to represent production-sequences; $\sigma(n)$ will represent the n^{th} production in the sequence σ

The sample space $\mathcal{S}$ for our experiment will be the set of all LMCD's under G. We will ignore the null derivation, so $\mathcal{S} \subseteq \Pi^+$. Given any $\rho \in \mathcal{S}$ such that ρ is a LMCD for $x \in L(G)$, then for $i = 1,2$, define

$$Pr(\rho|H_i) = \Gamma_i(x)$$

Since G is unambiguous, there is a one-to-one correspondence between points in $\mathcal{S}$ and strings over L(G), hence the fact that $\Gamma_i(.)$ is a true discrete probability measure implies that $Pr(.|H_i)$ is also a probability measure.

Suppose that $\rho = \rho(1)\rho(2)\rho(3)...\rho(n)$.
Then, by the unrestrictedness assumption, we have

$$Pr(\rho|H_i) = P_i(\rho(1))P_i(\rho(2))...P_i(\rho(n)) \qquad (1)$$

We now proceed to define certain compound events on the sample space $\mathcal{S}$ in which we have an interest. Consider the set of all production sequences which form initial segments of LMCD's. To define this set formally, let

$$\Sigma = \{\sigma \in \Pi^+ \mid \exists\, \omega \in \Pi^* \text{ such that } \sigma\omega \in \mathcal{S}\}$$

Each element of Σ may now be associated in a natural way with certain compound events, or subsets of $\mathcal{S}$. For all $\sigma \in \Sigma$, define $E_\sigma = \{\rho \in \mathcal{S} \mid \exists\, \omega \in \Pi^* \text{ such that } \sigma\omega = \rho\}$

Since the probability of a compound event is given by the sum of the probabilities of the sample points in it, we have (for i = 1,2 and all $\sigma \epsilon \sum$),

$$\Pr(E_\sigma|H_i) = \sum_{\rho \epsilon E_\sigma} \Pr(\rho|H_i) \qquad (2)$$

Combining (1) and (2) gives:

$$\Pr(E_\sigma|H_i) = \sum_{\rho \epsilon E_\sigma} P_i(\rho(1))P_i(\rho(2))\ldots P_i(\rho(n))$$

Noting that all production sequences in E_σ have the same initial segment $\sigma = \rho(1)\rho(2)\ldots\rho(k) = \sigma(1)\sigma(2)\ldots\sigma(k)$ followed by some sequence $\omega = \omega(1)\omega(2)\ldots\omega(r) = \rho(k+1)\ldots\rho(n)$ where k+r=n, we can write, pulling the constant factors out of the summation,

$$\Pr(E_\sigma|H_i) = P_i(\sigma(1))P_i(\sigma(2))\ldots P_i(\sigma(k)) \sum_{\{\omega|\sigma\omega \epsilon E_\sigma\}} P_i(\omega(1))\ldots P_i(\omega(r)) \qquad (3)$$

We now prove that, for any $\sigma \epsilon \sum$

$$\sum_{\{\omega|\sigma\omega \epsilon E_\sigma\}} P_i(\omega(1))\ldots P_i(\omega(r)) = 1 \qquad (4)$$

Once we have established this, (3) reduces to

$$\Pr(E_\sigma|H_i) = P_i(\sigma(1))P_i(\sigma(2))\ldots P_i(\sigma(k)) \qquad (5)$$

This is the centrally important multiplication rule for initial segments. Before a discussion of this rule, it remains to prove equation (4).

D. 2. Markov Chain Model

Consider the countably infinite Markov chain M whose states are represented by sentential forms in G. (See Feller(3) for a general reference on Markov Chains.) If m and m' are states of M, let the probability of a transition from m to m' be p where p is the probability under H_i associated with the production in π which transforms the sentential form represented by m into the sentential form represented my m' by expanding the left-most non-terminal. If no such production exists, then let the transition probability from m to m' be zero. Let all states corresponding to sentences of L(G) be absorbing with probability one. This is equivalent to assuming we have a 'null production rule' which is capable of transforming any sentence into itself with probability one. (In order to insure that this is a true Markov chain we are making use of the assumption that G_i is a proper stochastic grammar as defined in Appendix B.) Let the initial state of the Markov chain be the state corresponding to the sentential form consisting of the start symbol of G.

Let Y be the set of all state-sequences generated by this process. Let Y_T be the set of all sequences in Y in which all but a finite number of states are absorbing.

These state-sequences correspond to terminal sentences in L(G). Let $Y_N = Y - Y_T$. The consistency condition of G_1, when translated to this model, says that the probability of generating a sequence in Y_N is zero and the probability of generating a sequence in Y_T is one.

Let σ be an element of Σ and let α be the sentential form produced by applying σ to the start symbol of G. Let m_σ be the state of M corresponding to α. Now construct the Markov process M_σ which is identical to M in every respect except that m_σ is the start state. Let Y^σ, Y_N^σ and Y_T^σ be defined analogously to Y, Y_N, and Y_T.

We can now show that the probability that M_σ generates a sequence in Y_T^σ must be equal to one. Suppose this were not true. Then, since Y_N^σ and Y_T^σ are complementary in Y^σ, the probability that M_σ generates a sequence in Y_N^σ must be non-zero. Call this probability ξ. Now let β be the probability under M of generating a path from the start state of the original chain to m_σ. Let Z_N^σ be the set of all sequences in Y_N with σ as initial segment. Then the probability under M of generating a sequence in Z_N^σ is given by the product $\xi\beta$. Since Z_N^σ is a subset of Y_N, the probability of generating a sequence in Y_N is at least $\xi\beta$. This contradicts the consistency of G_1, hence our assumption was false, and we have established that the probability of generating a sequence in Y_T^σ by the Markov process M_σ is equal to one for any $\sigma \in \Sigma$. QED

In terms of our original generative grammar G_i, this means that with probability one, any initial segment of a derivation sequence will terminate in a finite number or steps. Thus equation (4) and hence equation (5) are established.

It should be noted that the multiplication rule

$$Pr(E_\sigma|H_i)=P_i(\sigma(1))P_i(\sigma(2))\ldots P_i(\sigma(k))$$

is in general valid only for sets of the form E_σ (i.e. corresponding to an initial segment of a LMCD) and does not imply any kind of independence among events associated with each production.

For example, the probability of the occurrence of π_j at the n^{th} position in a derivation is <u>not</u> given by $P_i(\pi_j)$. This latter probability represents the probability of the occurrence of π_j conditioned upon the existence of the premise of π_j in the sentential form upon which the production is applied. This condition is automatically fulfilled if we consider only initial segments of derivations. It is for this reason that a top-down parsing technique is essential.

D. 3. The SPRT

We now have a sample space $\mathcal{S}$ with a different probability measure under each hypothesis and a class of events $\{E_\sigma\}_{\sigma\in\Sigma}$ over $\mathcal{S}$. We have just established an efficient means of computing the probability associated with any E_σ. Assume now that either H_1 or H_2 is active,

but that we have no a priori knowledge about the likelihood of either hypothesis. We now perform a single random experiment whose outcome is completely described by exactly one point of $\mathcal{S}$. The experiment may be regarded as consisting of pushing a button and receiving a string x in return. The outcome of this experiment (a point in $\mathcal{S}$) specifies a finite family of events (subsets) of $\mathcal{S}$; namely all those events E_σ such that σ is an initial segment of the LMCD of the string x. Call this family of events $\mathcal{E}_x$. Note that this family is totally ordered by inclusion. That is to say, E_{σ_1} is contained within E_{σ_2} if σ_2 is an initial segment of σ_1.

The sequential discriminative parsing scheme is now used to examine each of the sets $E_\sigma \in \mathcal{E}_x$ in turn from largest to smallest. When either the family $\mathcal{E}_x$ is exhausted or certain stopping criteria are met, the algorithm decides that one of the hypotheses H_1 or H_2 is true and terminates.

The stopping condition is defined as follows: Before performing the experiment, select two numbers A and B (stopping boundaries). As we successively inspect each E_σ as described, we form the likelihood ratio

$$\lambda(E_\sigma) = \frac{\Pr(E_\sigma|H_1)}{\Pr(E_\sigma|H_2)}$$

and check if either $\lambda(E_\sigma) \geq A$ or $\lambda(E_\sigma) \leq B$. In the former case we decide H_1; in the latter case we decide H_2. We

shall see later that the discriminant grammar provides an effective means of recursively computing the likelihood ratio. (Actually, we will use the log of the likelihood ratio.) First we consider the determination and interpretation of A and B.

In the SPRT as developed by Wald, an unbounded number of observations is available and conditions are specified so that the likelihood ratio eventually reaches a stopping boundary with probability one. In the case of the sequential parsing scheme this is not true; it is quite possible that a parse may be completed before a boundary is reached. In this case we will make a maximum likelihood decision, i.e., we will decide H_1 if the log of the likelihood ratio is positive and decide H_2 otherwise. Because of this possibility of running out of features, the error analysis which follows differs somewhat from that given by Wald.

D. 4. Error Analysis

Let e_{21} represent the probability of deciding H_2 by encountering the boundary B when the true hypothesis is H_1. Let e_{12} represent the probability of deciding H_1 by encountering the boundary A when the true hypothesis is H_2. Let γ_i be the probability that a parse under H_i never reaches either boundary A or B. Assume now that at some point during the sequential decision scheme we have $\lambda(E_\sigma) \geq A$. For this E_σ we have

$$\lambda(E_\sigma) = \frac{Pr(E_\sigma|H_1)}{Pr(E_\sigma|H_2)} \geq A$$

By definition, $\Pr(E_\sigma|H_2) = e_{12}$ for this E_σ. Also, since under H_1 we must either terminate at A, terminate at B or fail to reach a boundary, we have

$$\Pr(E_\sigma|H_1) + e_{21} + \gamma_1 = 1 \tag{6}$$

Combining these facts yields

$$A \leq \frac{1-e_{21}-\gamma_1}{e_{12}} \tag{7}$$

Similarly, we can show that

$$B \geq \frac{e_{21}}{1-e_{12}-\gamma_2} \tag{8}$$

From these inequalities we can immediately derive upper bounds for the e_{ij}:

$$\begin{aligned} e_{12} &\leq 1/A \\ e_{21} &\leq B \end{aligned} \tag{9}$$

We can tighten the bounds (9) if we make some assumptions about the behavior of the sequence of likelihoods $\lambda(E_\sigma)$. One such assumption is that we will strive to have $e_{21} = e_{12}$. An additional, related assumption is that, under each hypothesis, the probability of classifying a sequence correctly using a maximum-likelihood decision rule is no less than the probability of classifying a sequence correctly under a pure sequential scheme where the $\lambda(E_\sigma)$ must cross a boundary before classification is made.

Several arguments can be made for the reasonableness of this assumption. First, if the decision boundaries are very far apart, there is a good chance that no decision at all will be made, whereas the maximum likelihood scheme always yields a decision. Secondly, it is natural to assume that with more observations available, a better decision can be made. This assumption is not reasonable, however, without the original condition that $e_{12} = e_{21}$ so that the total error is evenly distributed between both classes. If we let $\hat{e}$ be the maximum likelihood probability of error under both classes, The assumption says that

$$e_{12} + \gamma_2 \geq \hat{e}$$

and

$$e_{21} + \gamma_1 \geq \hat{e}$$

Substituting this into (7) and (8) and denoting $e_{21} = e_{12} = e$, we get

$$A \leq (1 - \hat{e})/e$$

and

$$B \geq e/(1 - \hat{e})$$

From which we get

$$e \leq [\min(B,1/A)](1 - \hat{e})$$

D. 5. Parsing Algorithm

We now relate the procedure described in this section in terms of the discriminant grammar. Assume that we

are given two commensurate stochastic grammars G_1 and G_2 and two stopping boundaries A and B as described above. Assume further that the common characteristic grammar of the given stochastic grammars is LL(k). Then construct discriminant grammar $D = LLRR(G_1,G_2)$ and set stopping boundaries $\alpha = \log A$ and $\beta = \log B$.

Then the sequential discriminative parsing scheme (SDPS) may be described as follows:

1. Initialize accumulator variable h to zero.
2. Find the next production π in top-down sequence using LL(k) techniques.
3. Let $h \leftarrow h + R(\pi)$
4. If $h \geq \alpha$, decide H_1 and stop.
5. If $h \leq \beta$, decide H_2 and stop.
6. If the parse is not finished, go to step 2.
7. If $h \leq 0$, decide H_2 and stop.
8. If $h > 0$, decide H_1 and stop.

This procedure is flow-charted in Figure 4.

EXAMPLE Suppose we are given two stochastic grammars

$$G_1 = (\{S,B\},\{c,g,f,h\}, \Pi ,P_1)$$

and

$$G_2 = (\{S,B\},\{c,g,f,h\}, \Pi ,P_2)$$

where Π, P_1 and P_2 are given in Table 1. Table 1 also gives $R(\pi_i)=\log P_1(\pi_i) - \log P_2(\pi_i)$. Under this definition of R, $D=LLRR(G_1,G_2) = (\{S,B\},\{c,g,f,h\}, \Pi ,R)$. Both

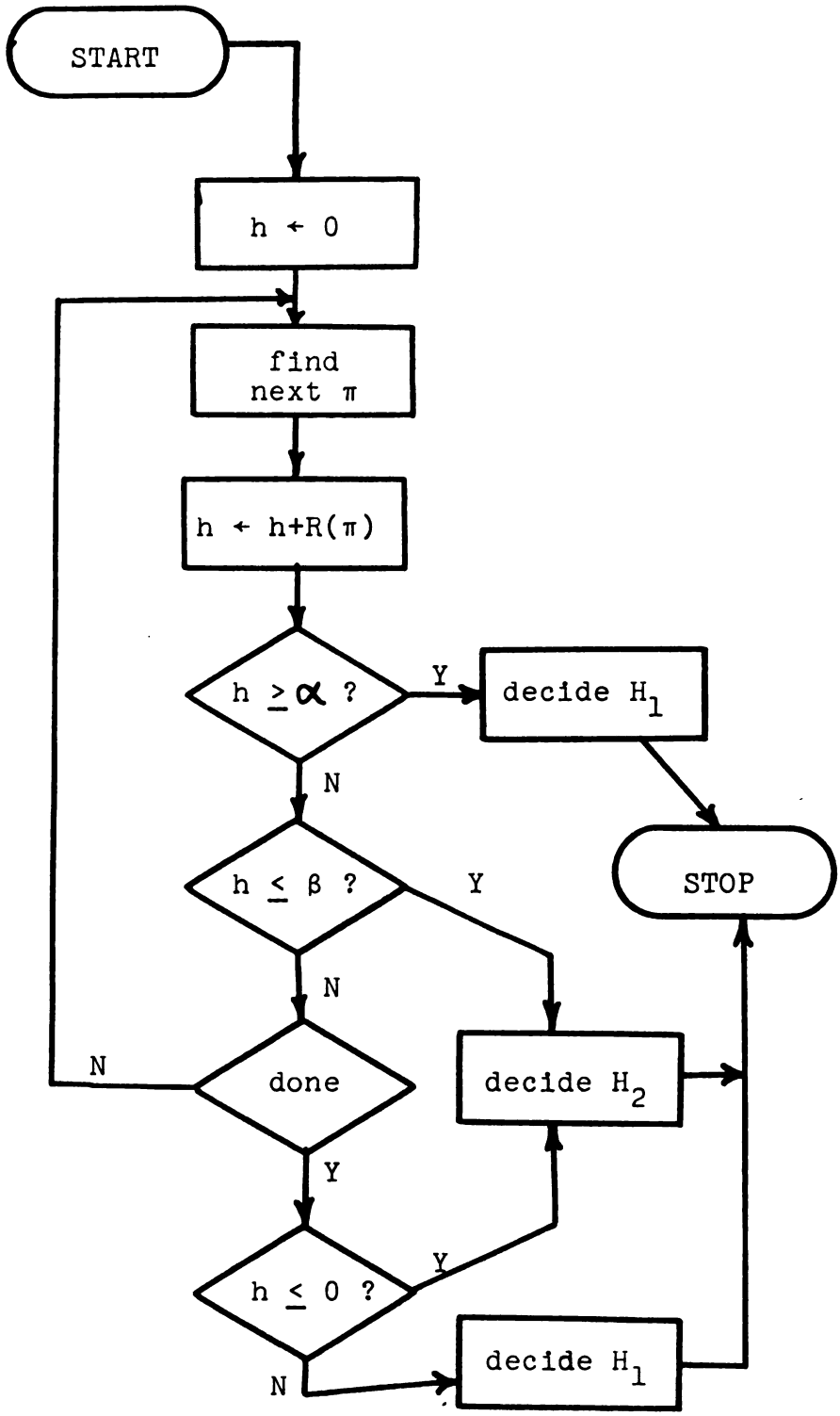


Figure 4 Sequential Discriminative Parsing Scheme

G_1 and G_2 are consistent and proper and their underlying characteristic grammar G is LL(1). Let us arbitrarily set A = 100 and B = .01. The probability of the parse reaching the wrong stopping boundary is then less than 1%. (Recall that this is not a bound on the overall error of the procedure.) We then have α = log 100 = 4.61 and β = log .01 = -4.61.

Suppose now that we are given the string ccccgfhhhhh. The underlying characteristic grammar belongs to a particularly simple class of LL(1) grammars for which we can determine one production of the LMCD for each symbol of the string scanned. Table 2 summarizes the results of the application of the SDPS to the given string. Recall that h is the accumulator variable representing the cumulative sum of $R(\pi_i)$. As indicated in the table, the parse was truncated and a decision was made that x was generated by G_1 after observing only four of the eleven symbols of the string. This was possible because the pre-determined upper stopping boundary 4.61 was exceeded.

Table 1 Two Stochastic Grammars and Their Log-Likelihood Ratio Representation

i	π_i	$P_1(\pi_i)$	$P_2(\pi_i)$	$R(\pi_i)$
1	S → cSB	.8	.2	1.39
2	S → g	.2	.8	-1.39
3	B → fBB	.3	.4	-.288
4	B → h	.7	.6	.154

Table 2 Example of the SDPS

Symbol Scanned	Production Determined	$R(\pi_1)$	h	Action
c	π_1	1.39	1.39	continue
c	π_1	1.39	2.78	continue
c	π_1	1.39	4.17	continue
c	π_1	1.39	5.56	stop; decide H_1

E. SUMMARY

In this chapter we have formulated the definition of the Discriminant Grammar and the decision regions which accompany it. We then demonstrated that the Discriminant Grammar has a natural application in providing a sufficient statistic for the two-class decision problem involving stochastic grammars. First a Bayesian approach was described and then it was shown how a sequential probability ratio test could be implemented using a top-down parsing technique. It should be noted that top-down parsing of an LL(k) language can be programmed quite readily using recursive descent.

In all the above statistical interpretations it was assumed that the two stochastic grammars involved are commensurate. If they are not commensurate, we cannot apply the above results, but all is not lost, since we can use training methods to be discussed later.

III. SOME LANGUAGE-THEORETIC RESULTS

A. REGULAR DISCRIMINANT GRAMMARS WITH RATIONAL COEFFICIENTS

A. 1. Informal Description

Discriminant grammars whose underlying characteristic grammars are regular have many interesting properties. We shall call such discriminant grammars regular discriminant grammars. In this section we intend to show that given any regular discriminant grammar $D = (V_N, V_T, \Pi, S, R)$ where all the $R(\pi_i)$ are rational, then for any rational number θ, the decision region $L^-(D,\theta)$ is a context-free language. The following informal argument will serve as a basis for a proof of this theorem:

Denote $R(\pi_i)$ by r_i, as before. Convert the numbers $\theta, r_1, r_2, \ldots r_n$ to integers by multiplying each by the least common multiple of their denominators. To simplify notation, we will retain the same names for these (now integral) numbers. In effect, we create a new discriminant grammar D with integral coefficients. This does not change the region $L^-(D,\theta)$. Now construct a non-deterministic finite-state automaton which accepts L(CHAR(D)). We shall now modify this into a non-deterministic push-down automaton which will accept $L^-(D,\theta)$. The existence of this

automaton implies that $L^{-}(D,\Theta)$ is a context-free language. We shall also show by example that $L^{-}(D,\Theta)$ is not necessarily regular.

We proceed as follows to construct the automaton: Let M be the maximum ab solute value of the (now integral) numbers $r_1, r_2, \ldots r_n, \Theta$. Clearly, we can construct a finite-state machine which is capable of adding any two integers of opposite sign and absolute value less than or equal to M and producing a result of absolute value less than or equal to M. Call this machine the "adder".

Next we need a push-down store, each location of which is capable of storing an encoding of an integer of absolute value less than or equal to M.

The final push-down automaton is an assemblage of these three components (non-deterministic recognizer, adder, push-down store) plus some mechanism for integrating these parts, which works as follows: Suppose the recognizer (in a non-deterministic mode) makes a transition which indicates that π_i is in the derivation of the input string. If the push-down store is empty, the operation of the machine would be to place r_i on the store. If the store is not empty, denote the value of the top element by K. If r_i and K are of the same sign, then the machine should push r_i onto the stack. If r_i and K differ in sign, then the machine should add r_i and K, forming a number r' of lesser absolute value than either r_i or K, pop the stack, exposing a new top symbol K', and then compare r'

with K'. The machine should proceed in this way until r' is eventually added into the stack. This stack manipulation algorithm has two very important properties:

1) At no time is a number with absolute value greater than M on the stack.

2) All numbers on the stack have the same sign.

When the recognizer reaches a final state, the adder is then used in exactly the same way to add θ into the stack. After this operation, the input string is accepted as an element of $L^{-}(D,\theta)$ if and only if the top stack element is negative.

This informal description of the operation of the machine is flow-charted in Figure 5 and Figure 6. The multiple arrows in Figure 5 denote the non-deterministic step in which the non-deterministic finite-state automaton discovers another step in the parse. Figure 6 defines the procedure ADDSTK with formal parameter r. In both figures, K refers to the top element of the stack. NDFA stands for non-deterministic finite-state automaton In Figure 6 the condition "K*r > 0" does not imply that an actual multiplication need be done, only that K and r are non-zero and have the same sign.

A. 2. Detailed Construction

We now give a detailed formal construction to implement the foregoing informal description. A (non-deterministic) push-down automaton (PDA) is a 7-tuple

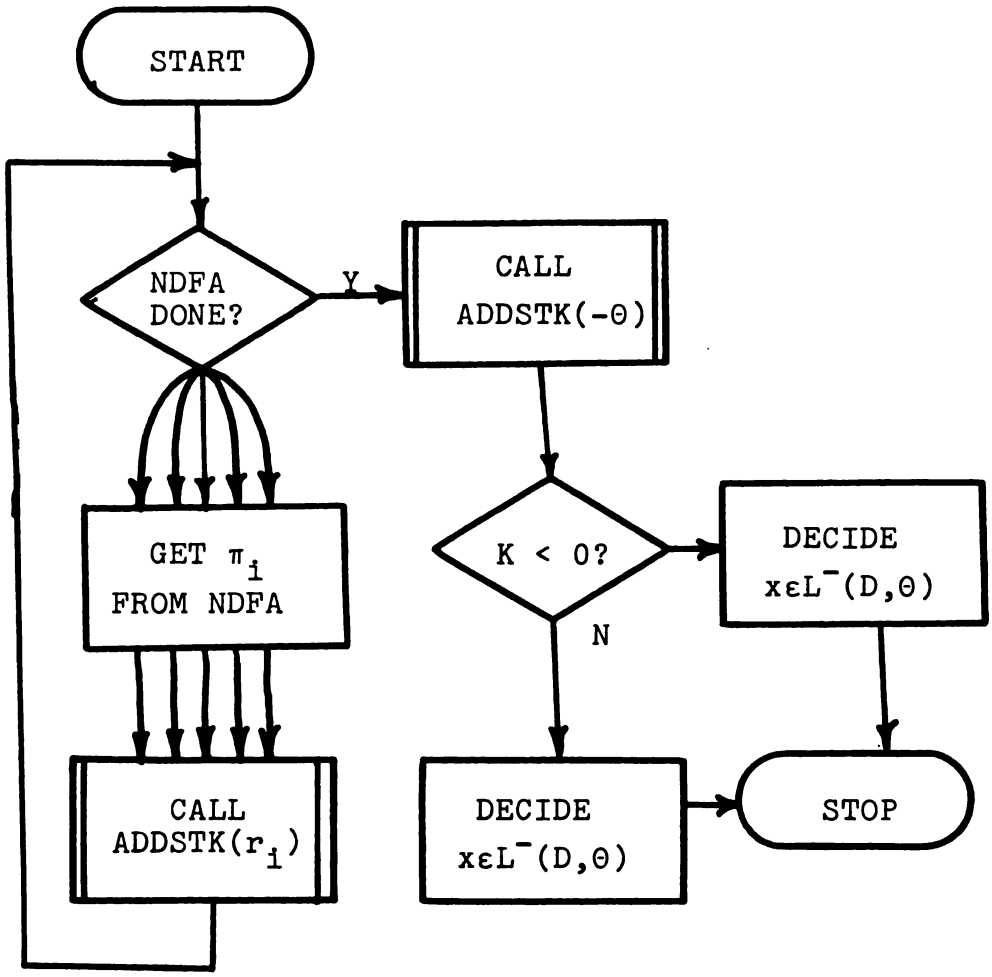


Figure 5 Flowchart of Push-Down Automaton

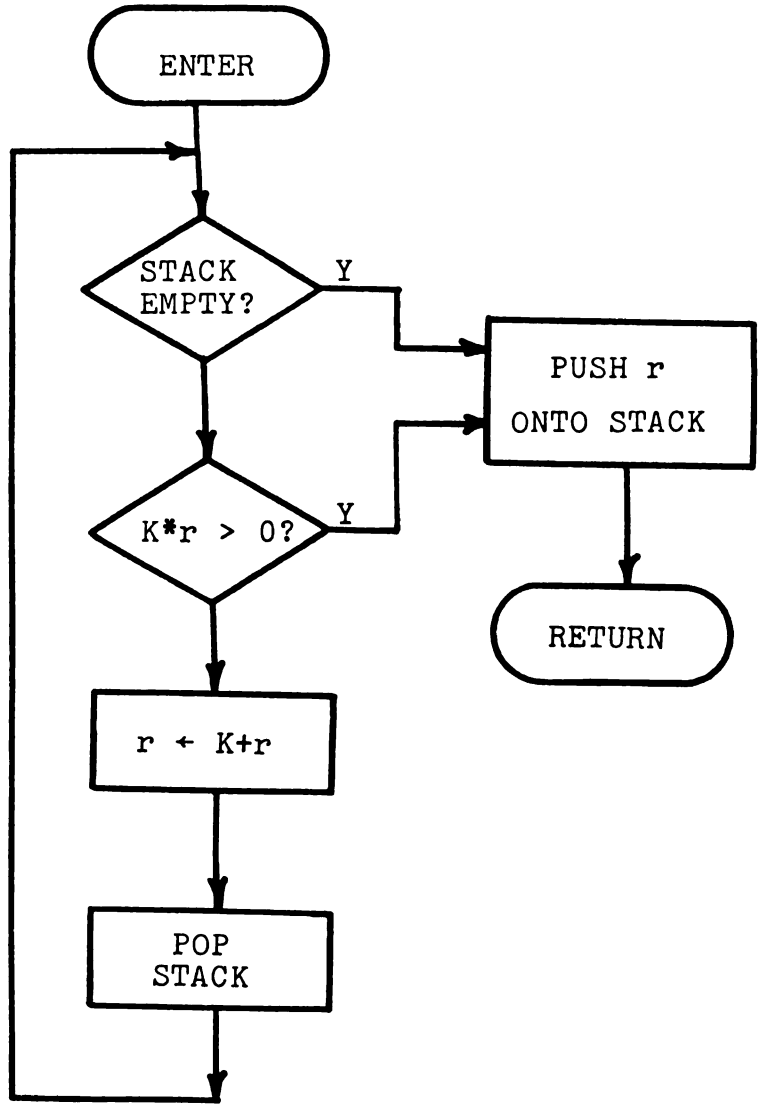


Figure 6 Procedure ADDSTK(r)

$$M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where:

K is a finite set of states

Σ is a finite input alphabet

Γ is a finite stack alphabet

δ is a function from $K\times(\Sigma \cup \{\varepsilon\})\times\Gamma$ into $\mathcal{P}(K\times\Gamma^*)$

$q_0 \varepsilon K$ is the initial state

$Z_0 \varepsilon \Gamma$ is the initial stack symbol

$F \subseteq K$ is a set of final states

The details of the operation of a PDA may be found in Hopcroft and Ullman(10). The fundamental result we will use is the fact that a language is accepted by a non-deterministic push-down automaton if and only if the language is context free.

We will begin with a regular discriminant grammar with rational coefficients and a rational number Θ and we will construct a PDA which will accept $L^-(D,\Theta)$. First, as before, we convert all coefficients, including Θ, to integers in the range -M to +M. This does not alter the region $L^-(D,\Theta)$.

Let $D = (V_N, V_T, \Pi, S=B_1, R)$ where

$V_N = \{B_i\}$ and $V_T = \{a_i\}$

Let A, A', and T be three symbols not in V_N.

Then let

$$K = (V_N \cup \{A, A', T\}) \times \{-M, -M+1, \ldots 0 \ldots M-1, M\}$$

$$\Sigma = V_T$$

$$\Gamma = \{-M, -M+1, \ldots -1, 1, 2, \ldots M-1, M, Z_0\}$$

$$q_0 = (S, 0)$$

$$F = \{(T, 0)\}$$

The construction of the transition function δ is somewhat more complex. For each non-terminal B_i and terminal a_k there is a (possibly empty) set of production

$$\{B_i \rightarrow a_k B_j\}_{j=1}^{\ell} \tag{1}$$

and possibly a production of the form

$$B_i \rightarrow a_k \tag{2}$$

We will use the symbol C to represent either non-terminals (B_j) or either of the special symbols A or A'. Specifically, the set designated $\{C_{ik}^j\}$ contains the ℓ symbols $\{B_j\}_{j=1}^{\ell}$ as represented by (1) and also the symbol A if there is a production of the form (2). (In this case the C_{ik}^j can be considered to have the superscript $\ell+1$.) Analogously, the set $\{n_{ik}^j\}$ is the set of integers (r's) associated with the productions (1) and (2).

For example, suppose that the productions of π having B_i on the left and a_k on the right and their associated r-values are:

Production	r-value
$B_i \rightarrow a_k B_1$	+5
$B_i \rightarrow a_k B_2$	-2
$B_i \rightarrow a_k$	0

Then $\{C_{ik}^j\}_{j=1}^3 = \{B_1, B_2, A\}$ and

$\{n_{ik}^j\}_{j=1}^3 = \{5, -2, 0\}$

In the following, m_{ik} represents the number of productions in Π with B_i on the left and a_k on the right. To avoid subscripted superscripts, we will usually omit the subscripts on m when context makes them clear. Z represents an arbitrary stack symbol (either a number in the specified range or the stack start symbol Z_0). We now have sufficient notation to define δ.

First, for each B_i, δ should contain the following non-deterministic, non-ε transition rules (for all $Z \varepsilon \Gamma$):

(1) $$\delta((B_i,0),a_k,Z) = \{((C_{ik}^j, n_{ik}^j), Z)\}_{j=1}^m$$

Informally, this is equivalent to saying that no matter what is on the stack, if the machine is in state $(B_i,0)$ and a_k is read, then, for each production π of the form $B_i \rightarrow a_k B_j$ with $R(\pi)=n$, the machine moves (non-deterministically) into state (B_j,n). If there is a production π of the form $B_i \rightarrow a_k$ with $r(\pi)=n$, then the machine also moves into state (A,n). In all cases the stack symbol remains unchanged. The purpose of this class of instructions is

to allow the machine to recognize when a production rule might have been applied and to incorporate the r-value of that production rule into the memory of the PDA's finite state control.

Next we need a mechanism to transfer these r-values from the finite-state control onto the push down stack as described informally earlier. For each C, therefore, we shall include the following ε-rules: For all combinations of non-zero n and Z such that n and Z have the same sign or $Z=Z_0$, include the rules

(2) $\delta((C,n),\varepsilon,Z) = \{((C,0),nZ)\}$

i.e., if the number in the finite-state control has the same sign as the number on top of the stack, or if the symbol on top of the stack is Z_0, then the number from the finite-state control may be pushed onto the stack.

For all combinations of n and Z such that n and Z differ in sign and n is not zero, include the rules:

(3) $\delta((C,n),\varepsilon,Z) = \{((C,Z+n),\varepsilon)\}$

i.e., if the number in the finite state control differs in sign from the number on top of the stack, we add the top stack element into the finite state control and pop the stack.

By using the above rules, we eventually arrive in the state (A,0) if and only if there is a derivation for the input string in CHAR(D). The next step is to put -0 into

the finite control. This may be accomplished by the single production:

(4) $\delta((A,0),\varepsilon,Z) = \{((A',-\theta),Z)\}$

for all $Z\varepsilon\,\Gamma$. Previously defined transition rules may now be used to transfer $-\theta$ from the finite control to the stack, ultimately sending the machine into state $(A',0)$. The additional symbol A' is necessary here to insure that this rule is used only once. Now it remains only to interrogate the top stack symbol and send the machine into the final state if and only if this top symbol is negative. This is accomplished by including the following set of productions for all Z less than zero:

(5) $\delta((A',0),\varepsilon,Z) = \{((T,0),Z)\}$

The existence of the above construction establishes the following:

THEOREM Given a regular discriminant grammar D with rational values for $R(\pi_1)$ and given any rational number θ, the language $L^-(D,\theta)$ is context-free.

Simple modifications to the above construction could be made to show that $L^+(D,\theta)$ and $L^0(D,\theta)$ are also context-free.

A. 3. Example

As an example, consider the following regular discriminant grammar with rational coefficients:

$$D = (\{S=B_1\},\{a_1,a_2,a_3\}, \Pi ,S,R)$$

Where Π and R are given by

π_i	r_i
$B_1 \rightarrow a_1B_1$	+1
$B_1 \rightarrow a_2B_1$	-1
$B_1 \rightarrow a_3$	+1

Clearly, $L^-(D,1) = \{x | x\varepsilon\{a_1,a_2\}^* a_3$ and $x(a_2) > x(a_1)\}$ where the notation x(a) means the number of occurrences of the symbol a in the string x. Since this language is not regular, this example shows that we cannot strengthen the theorem to say that $L^-(D,\theta)$ must be regular.

According to the construction,

$$M = (K, \Sigma , \Gamma ,\delta, q_0,Z_0,F) \quad \text{where}$$

$$K = \{(B_1,1),(B_1,0),(B_1,-1),(A,1),(A,0),$$
$$(A,-1),(A',1),(A',0),(A',-1),(T,1),$$
$$(T,0),T,-1)\}$$

$$\Sigma = \{a_1,a_2,a_3\}$$

$$\Gamma = \{1,-1,Z_0\}$$

$$q_0 = (B_1,0)$$

$$F = \{(T,0)\}$$

and the transition function δ is given in Table 3. In that table, each element of the transition function is labelled I.J. where the I refers to one of the five rules used to generate it and the J is simply an ordinal number within a group.

Table 4 gives a trace of the action of this automaton in accepting the string $x=a_2a_1a_2a_3$. Following the convention of Hopcroft and Ullman, the bottom of the stack is on the right and stack symbols are added or deleted from the left. Also remember that "-1" is a single stack symbol, not two. As expected, this string is accepted because the PDA finally enters state (T,0).

Table 3 PDA Transition Function (Example)

1.1 $\delta((B_1,0),a_1,-1) = \{((B_1,1),-1)\}$

1.2 $\delta((B_1,0),a_1,1) = \{((B_1,1),)\}$

1.3 $\delta((B_1,0),a_1,Z_0) = \{((B_1,1),Z_0)\}$

1.4 $\delta((B_1,0),a_2,Z_0) = \{((B_1,-1),Z_0)\}$

1.5 $\delta((B_1,0),a_2,-1) = \{((B_1,-1),-1)\}$

1.6 $\delta((B_1,0),a_2,1) = \{((B_1,-1),1)\}$

1.7 $\delta((B_1,0),a_3,Z_0) = \{((A,1),Z_0)\}$

1.8 $\delta((B_1,0),a_3,-1) = \{((A,1),-1)\}$

1.9 $\delta((B_1,0),a_3,1) = \{((A,1),1)\}$

2.1 $\delta((B_1,1),\varepsilon,Z_0) = \{((B_1,0),1Z_0)\}$

2.2 $\delta((B_1,-1),\varepsilon,Z_0) = \{((B_1,0),-1Z_0)\}$

2.3 $\delta((A,1),\varepsilon,Z_0) = \{((A,0),1Z_0)\}$

2.4 $\delta((A,-1),\varepsilon,Z_0) = \{((A,0),-1Z_0)\}$

2.5 $\delta((A',1),\varepsilon,Z_0) = \{((A',0),1Z_0)\}$

2.6 $\delta((A',-1),\varepsilon,Z_0) = \{((A',0),-1Z_0)\}$

Table 3 (continued)

2.7 $\delta((B_1,1),\varepsilon,1) = \{((B_1,0),11)\}$

2.8 $\delta((B_1,-1),\varepsilon,-1) = \{((B_1,0),-1-1)\}$

2.9 $\delta((A,1),\varepsilon,1) = \{((A,0),11)\}$

2.10 $\delta((A,-1),\varepsilon,-1) = \{((A,0),-1-1)\}$

2.11 $\delta((A',1),\varepsilon,1) = \{((A',0),11)\}$

2.12 $\delta((A',-1),\varepsilon,-1) = \{((A',0),-1-1)\}$

3.1 $\delta((B_1,1),\varepsilon,-1) = \{((B_1,0),\varepsilon)\}$

3.2 $\delta((B_1,-1),\varepsilon,1) = \{((B_1,0),\varepsilon)\}$

3.3 $\delta((A,1),\varepsilon,-1) = \{((A,0),\varepsilon)\}$

3.4 $\delta((A,-1),\varepsilon,1) = \{((A,0),\varepsilon)\}$

3.5 $\delta((A',1),\varepsilon,-1) = \{((A',0),\varepsilon)\}$

3.6 $\delta((A',-1),\varepsilon,1) = \{((A',0),\varepsilon)\}$

4.1 $\delta((A,0),\varepsilon,Z_0) = \{((A',-1),Z_0)\}$

4.2 $\delta((A,0),\varepsilon,1) = \{((A',-1),1)\}$

4.3 $\delta((A,0),\varepsilon,-1) = \{((A',-1),-1)\}$

5.1 $\delta((A',0),\varepsilon,-1) = \{((T,0),-1)\}$

Table 4 Trace of PDA (Example)

input	δ-rule	state	stack contents
		$(B_1,0)$	Z_0
a_2	1.4	$(B_1,-1)$	Z_0
ε	2.2	$(B_1,0)$	$-1Z_0$
a_1	1.1	$(B_1,1)$	$-1Z_0$
ε	3.1	$(B_1,0)$	Z_0
a_2	1.4	$(B_1,-1)$	Z_0
ε	2.2	$(B_1,0)$	$-1Z_0$
a_3	1.8	$(A,1)$	$-1Z_0$
ε	3.3	$(A,0)$	Z_0
ε	4.1	$(A',-1)$	Z_0
ε	2.6	$(A',0)$	$-1Z_0$
ε	5.1	$(T,0)$	$-1Z_0$

B. REGULAR DISCRIMINANT GRAMMARS WITH UNRESTRICTED COEFFICIENTS

In this section we show that if the r_i and θ are not constrained to be rational, then there are decision regions $L^{-}(D,\theta)$ which are not context-free.

THEOREM The class of languages expressible in the form $L^{-}(D,\theta)$, where CHAR(D) is regular, is uncountable.

PROOF For each real number ϕ, let $D(\phi)$ represent the following discriminant grammar:

$$D(\phi) = (\{S\},\{a,b,c\}, \Pi ,S,R)$$

where Π and R are given by

i	π_i	$R(\pi_i)$
1	$S \rightarrow aS$	$\cos(\phi)$
2	$S \rightarrow bS$	$-\sin(\phi)$
3	$S \rightarrow c$	0

ϕ may be regarded as a parameter of the discriminant grammar. We now define the uncountable class of languages.

$$\{L^{-}(D(\phi),0) \mid 0 \leq \phi \leq \pi/2\}$$

We now show that the languages in this class are distinct. As before, let x(a) denote the number of occurrences of the symbol a in the string x. Then we can express

$$L^{-}(D(\phi),0) = \{x\varepsilon\{a,b\}^{*}c \mid x(a)\cos(\phi)-x(b)\sin(\phi)<0\}$$

$$= \{x\varepsilon\{a,b\}^{*}c \mid x(a)/x(b)<\tan(\phi)\}$$

Now consider two languages

$$L^{-}(D(\phi_1),0) \quad \text{and} \quad L^{-}(D(\phi_2),0)$$

where $\phi_1<\phi_2$.

Now choose two integers I and J such that

$$\tan(\phi_1)<I/J<\tan(\phi_2).$$

Then the string $a^{I}b^{J}c$ is an element of $L^{-}(D(\phi_2),0)$ but is not an element of $L^{-}(D(\phi_1),0)$, hence the two languages are distinct.

QED

Since the class of context-free languages is countable, it follows immediately that

COROLLARY There exist non-context-free languages of the form $L^{-}(D,\theta)$ where CHAR(D) is regular.

This statement can be strengthened by substituting the phrase "recursively enumerable" for "context-free".

C. SUMMARY

In this chapter we have shown that if we have a regular discriminant grammar with rational coefficients, the resulting decision regions are context-free but not necessarily regular. On the other hand, if the coefficients are not constrained to be rational, there are uncountably many decision regions expressible as $L^{-}(D,\theta)$ hence there are decision regions which are not recursively enumerable.

The latter result is not surprising and there is an analogous result for probabilistic automata. The first-mentioned theorem is of much greater theoretical interest and provides a new characterization of context-free languages.

IV. DISCRIMINANT GRAMMARS AND PROBABILISTIC AUTOMATA

In this section we consider the relation between a decision region of a regular discriminant grammar and the language defined by a probabilistic automaton. A probabilistic automaton (PA) may be defined as a quintuple

$$M = (K, \Sigma, I, F, \{A(a): a \varepsilon \Sigma\})$$

where

$K = \{k_i\}_{i=1}^{n}$ is a finite set of states

Σ is a finite set of input symbols

I is an n-component stochastic row vector (The "initial distribution")

F is an n-component column vector consisting of 0's and 1's. (The final-state vector)

$\{A(a): a \varepsilon \Sigma\}$ is a set of n by n stochastic (sum along any row=1) transition matrices, one for each input symbol.

M defines a mapping from Σ^* into the reals in the following way: Given any string $x = a_1 a_2 a_3 \ldots a_r \varepsilon \Sigma^*$,

Let A(x) be defined as

$$A(x) = \prod_{i=1}^{r} A(a_i)$$

Then the real valued "probabilistic event" $E_M(x)$ is defined for all $x \in \Sigma^*$ as

$$E_M(x) = IA(x)F$$

Finally, given any PA M and a real number λ (a "cutpoint"), define the language (probabilistic cut-point event) accepted by M with cutpoint λ as:

$$T(M,\lambda) = \{x \mid x \in \Sigma^* \text{ and } E_M(x) > \lambda\}$$

Paz(19) shows that this class of languages is not changed if we remove the restrictions that I and the transition matrices be stochastic and if we allow F to be an arbitrary vector. In this case we use the terminology "pseudo-probabilistic automaton" (PPA) and "pseudo-probabilistic event". The language accepted by a PPA with cutpoint λ is called a "general cut-point event" (GCE).

We will now show that, given any regular discriminant grammar D, the region $L^+(D,\theta)$ is a GCE. Suppose that we are given a regular discriminant grammar $D=(V_N,V_T,\Pi,S,R)$. Since D is regular, all productions must be of the form

$$B_i \rightarrow a_k B_j$$

or

$$B_i \rightarrow a_k$$

In the former case denote the associated r-value as r^i_{kj} and in the latter case as r^i_{k0}.

Given any cutpoint θ, we will now construct a PPA M and a cutpoint λ such that $L^+(D,\theta) = T(M,\lambda)$. Let $K = V_N \cup \{T\}$, where T is some symbol not already in V_N which will serve as an "acceptance state". Denote the cardinality of K by n, so that $K = \{S=B_1, B_2, B_3, \ldots B_n=T\}$. Let the set of input symbols for M be identical to the set of terminal symbols of D ($\Sigma = V_T$). Let I be the n-component row vector with a 1 in the first position and 0's elsewhere. F is defined as the n-component column vector with a one in the n^{th} position and zeros elsewhere. Finally we construct the set of matrices $\{A(a): a \in \Sigma\}$. The essential construction here involves exponentiation so that the additive structure of the discriminant grammar may be translated to the essentially multiplicative structure of the PPA. To accomplish this, for each production of the form $B_i \rightarrow a_k B_j$ in Π, let the (i,j) element of $A(a_k)$ be $\exp(r^i_{k0})$. Let all other entries of the matrices be 0. (Note that the entire last row of each matrix is zero.) Finally, let $\lambda = \exp(\theta)$. We now show that the above PPA with cutpoint λ accepts all strings in $L^+(D,\theta)$ and that these are the only strings which it accepts.

Given any string x in $L^+(D,\theta)$, there must be exactly one derivation for that string in CHAR(D). Let this derivation be given by

$$S = B_1 \Rightarrow a_1B_2 \Rightarrow a_1a_2a_3 \Rightarrow \ldots \Rightarrow a_1a_2\ldots a_r = x$$

where the productions applied are $\pi_1,\pi_2,\ldots,\pi_r$. Since $x\varepsilon L^+(D,\Theta)$, we must also have

$$\sum_{i=1}^{r} R(\pi_i) > \Theta$$

We must now show that

$$I\left(\prod_{k=1}^{r} A(a_k)\right) F < \lambda$$

That is to say, the (1,n) element of the matrix

$$\prod_{k=1}^{r} A(a_k) \qquad \text{must exceed } \lambda \ .$$

Let us decompose this matrix into a product of two matrices by choosing some p such that $1 \leq p < r$. Then

$$\prod_{k=1}^{r} A(a_k) = \left(\prod_{k=1}^{p} A(a_k)\right) \left(\prod_{k=p+1}^{r} A(a_k)\right)$$

Call the matrix on the left U and the two matrices on the right G and H respectively. Then by the definition of matrix multiplication,

$$U_{1n} = \sum_{i=1}^{n} G_{1i}H_{in}$$

If more than one term in the above sum is non-zero, this would imply that there is more than one distinct derivation in CHAR(D) for x. Since CHAR(D) is unambiguous, we can say that only one of the products $G_{li}H_{in}$ is non-zero. Call this product $G_{ls}H_{sn}$. By decomposing G and H in exactly the same way, we can show that G_{ls} and H_{sn} are simple products rather than sums of products. Continuing in this way, we can show that

$$I\left(\prod_{k=1}^{r} A(a_k)\right) F = \prod_{k=1}^{r} \rho_k$$

where each ρ_k is some element of $A(a_k)$. If π is the p^{th} production used in the derivation of x, and π is $B_i \rightarrow a_k B_j$ then ρ_m must be $\exp(r^i_{kj})$. Thus we have shown that

$$E_M(x) = \prod_{k=1}^{r} (\exp(R(\pi_k)))$$

or, equivalently

$$E_M(x) = \exp \sum_{k=1}^{r} R(\pi_k)$$

Since we are assuming that

$$\sum_{k=1}^{r} R(\pi_k) > \Theta$$

and

$$\lambda = \exp(\Theta).$$

The monotonicity of the exponential function gives us the result that $E_M(x)>\lambda$. Thus we have shown that

$$x\varepsilon L^{+}(D,\theta) \Rightarrow x\varepsilon T(M,\lambda)$$

under the given construction. To establish the converse, we must consider two cases:

1) $x\varepsilon L(CHAR(D))$

2) $x\notin L(CHAR(D))$

In the first case, x has a derivation in CHAR(D) and, as above, we can establish that

$$E_M(x) = \exp \sum_{k=1}^{r} R(\pi_k)$$

Since we are assuming now that $E_M(x)>\lambda$ and that $\lambda = \exp(\theta)$,

$$\sum_{k=1}^{r} R(\pi_k)>\theta$$

or, equivalently, that $x\varepsilon L^{+}(D,\theta)$.

The second case, $x\varepsilon L(CHAR(D))$ cannot occur, since in this case no derivation for x in CHAR(D) exists, hence

$$\prod_{k=1}^{r} A(a_k) = 0$$

contradicting the assumption that $E_M(x)>\lambda$. (Recall that $\lambda = \exp(\theta)$, hence is always positive.)

Invoking the equivalence between PA and PPA, we have now proved the following:

THEOREM Given any regular discriminant grammar D and cut-point Θ, there exists a PA M and a cutpoint λ such that $L^{+}(D,\Theta) = T(M,\lambda)$.

Consider the following example:

Let $D = (\{S=B_1,B_2\},\{a_1,a_2,a_3\},\Pi,S,R)$ where Π and R are given as follows:

π_i			$R(\pi_i)$
B_1	$\rightarrow$	a_1B_2	1
B_2	$\rightarrow$	a_2B_1	0
B_1	$\rightarrow$	a_3B_1	-1
B_1	$\rightarrow$	a_3	-1

Then $L(CHAR(D)) = ((a_1a_2)^{*}a_3^{*})^{*}a_3$ and $L^{+}(D,0) = \{x \mid x\varepsilon L(CHAR(D)) \text{ and } x(a_1) > x(a_3)\}$ where $x(a_i)$ denotes the number of occurrences of a_i in x. Then the pseudo-probabilistic automaton is constructed as follows:

$M = (K, \Sigma, I, F, \{A(a): a\varepsilon\Sigma\})$ where

$K = \{S=B_1,B_2,T\}$

$\Sigma = \{a_1,a_2,a_3\}$

$I = (1,0,0)$

$F = (0,0,1)^{T}$

$$A(a_1) = \begin{pmatrix} 0 & e & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A(a_2) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A(a_3) = \begin{pmatrix} e^{-1} & 0 & e^{-1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and the generalized cut-point event equivalent to $L^+(D,0)$ is $T(M,1)$. Let us check some sample strings:

1) $x = a_1a_2a_1a_2a_3$ $\quad (x\varepsilon L^+(D,0))$

$$E_M(x) = (1,0,0)A(a_1)A(a_2)A(a_1)A(a_2)A(a_3)(0,0,1)^T$$

$$= (1,0,0) \begin{pmatrix} e & 0 & e \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = e$$

since $e > 1$, $x\varepsilon T(M,1)$.

2) $x = a_1a_2a_3$ $\quad (x\varepsilon L(CHAR(D))$ but $x\notin L^+(D,0))$

$$E_M(x) = (1,0,0)A(a_1)A(a_2)A(a_3)(0,0,1)^T$$

$$= (1,0,0) \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 1$$

hence $x\notin T(M,1)$.

3) $x = a_1 a_3$ $\quad (x \varepsilon L(CHAR(D)))$

$$E_M(x) = (1,0,0)A(a_1)A(a_3)(0,0,1)^T$$

$$= (1,0,0) \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 0$$

hence, as predicted by the theorem, $x \notin T(M,1)$.

V. SUMMARY AND RECOMMENDATIONS

A. SUMMARY

This thesis introduces the idea of a discriminant grammar as a tool for syntactic pattern recognition, explores certain properties which derive from the definition, and explains certain techniques which the discriminant grammar makes possible.

Chapter I provides a motivation for the discriminant grammar concept and provides some informal examples of the advantages of discriminant grammars.

Chapter II gives the formal definition of discriminant grammars and goes on to show how they may be used to provide a sufficient statistic for the two-class decision problem involving stochastic languages. Specific formulations are given for the Bayesian case and the SPRT. The existence of the Sequential Discriminative Parsing Scheme is one of the principal contributions of this thesis. This chapter also gives a technique for the calculation of bounds on the Bayes error for linear grammars.

Chapter III proves some results on the nature of the decision regions of regular discriminant grammars. The result of greatest theoretical interest is the theorem which says that decision regions of regular discriminant grammars with rational coefficients are context free.

Chapter IV provides the proof of a result which establishes a connection between the regular discriminant grammar and probabilistic automata.

Finally, some suggestions are given for future work involving the learning of discriminant grammar coefficients from labeled empirical samples.

B. TRAINING METHODS

Insofar as pattern recognition is intended as a practical art, it must incorporate a learning or inductive phase as well as a decision algorithm. The decision schemes presented in this thesis center around the context-free unambiguous discriminant grammar. It will be necessary to develop schemes for the inference of such a discriminant grammar given some kind of empirical sample of strings from the classes between which we wish to discriminate. Two distinct problems arise -- inference of the structural component and inference of the numerical component of the discriminant grammar. The former is a far more difficult problem than the latter; in fact, the latter sort of learning is usually designated by a less imposing word than "inference" such as "training".

In general, the inference of a discriminant grammar is a two-stage process: determination of the structural component (the characteristic grammar) followed by the determination of the numerical component (The function R and the cutpoint θ). It is the purpose of this section to demonstrate the following two points:

1) Once the structural component has been determined, the determination of the numerical component is mathematically trivial.

2) The existence of the numerical component makes the determination of the structural component far less critical to the performance of the decision algorithm.

In the inference of ordinary regular or context-free phrase structure grammars (see Fu and Booth(8) for a survey), it is typical to have two finite empirical samples of strings, say L^+ and L^-, and be required to exhibit a grammar which accepts L^+ but does not accept L^-. The exponential combinatorics of known methods for accomplishing this make the task infeasible for samples of any substantial size, nor is it known how to make effective use of a priori human knowledge about apparent differences between the sample sets. In addition, the corruption of a single string by noise may sabotage the entire algorithm.

In contrast, let us now consider the inference problem in terms of discriminant grammars. Suppose we are given two sample sets L^+ and L^- and we are asked to exhibit a discriminant grammar D and cutpoint θ such that $L^+ \subseteq L^+(D,\theta)$ and $L^- \subseteq L^-(D,\theta)$. As a first step, we must determine a phrase structure grammar G with the property that $L(G) \supseteq L^+ \cup L^-$ which will serve as a characteristic grammar for D. There are a great many obvious grammars which will satisfy this requirement and we are free to use our human powers of inference to select grammars which seem like good candidates. The primary criterion in which we are interested is that G have some productions

which will be of most use in generating strings from L^+ and some productions more useful in generating strings from L^-. If we wish to allow for noisy strings, we can choose G such that L(G) is as large as possible, in fact, we may well choose G such that $L(G) = V_T^*$.

Once we have chosen the characteristic grammar for D (or, more probably, several candidate characteristic grammars), we are now ready for the second (numerical) stage of the inference process. To accomplish this, we first determine the structural indices (using the mapping S defined previously) of all strings in L^+ and L^-. The problem of numerical training is now simply one of finding a separating hyperplane between these two sets of indices. If these sets happen to be linearly separable, we have made a good choice of G and the desired coefficients may be found by means of the perceptron algorithm. In the more likely case that the sets of indices are not linearly separable, (or in the case where the sample strings are probabilistically generated infinite sequences), we may find coefficients which are in some sense "good" by using well-known variants of the perceptron algorithm. (See Duda & Hart(4)).

As a trivially simple example, consider the following two sample sets:

L^+ = {abbba , babba , bb}

and

$L^- = \{aaa , abaaa , abbaaa , a\}$

We note that the strings in L^- contain more a's than b's and vice versa for the strings in L^+. We therefore construct the following tentative characteristic grammar G:

$G = (\{S\},\{a,b\}, \Pi ,S)$

Where Π is given by

1) $S \rightarrow aS$

2) $S \rightarrow bS$

3) $S \rightarrow \epsilon$

The structural indices of the sample strings may be summarized as follows:

STRING	CLASS	S(x)
abbba	+1	(2,3,1)
babba	+1	(2,3,1)
bb	+1	(0,2,1)
aaa	-1	(3,0,1)
abaaa	-1	(4,1,1)
abbaaa	-1	(4,2,1)
a	-1	(1,0,1)

The two classes are indeed linearly separable. A suitable set of coefficients might be (-1,+1,0) with threshold 0. In terms of the discriminant grammar we are seeking this gives us $r_1 = -1$, $r_2 = +1$, $r_3 = 0$ and $\theta = 0$.

Although this problem is admittedly trivial, the point is that it is usually far easier to separate L^+ from L^- than it is to analyze the structure of either L^+ or L^-, thereby avoiding complex problems of structural inference, (or at least deferring the structural inference problem until the complexity of the problems rise to meet the capabilities of the analytical technique).

It is hoped that investigation of the various aspects of this training problem will provide fruitful ground for further research.

C. OTHER FUTURE WORK

There are several other directions in which this work might be extended. First, an improved error analysis for the sequential discriminative parsing scheme would be desirable, including estimates of the amount of work saved by truncating the parse. It would also be valuable to see how the new field of grammatical inference could be brought to bear on the problem of constructing characteristic grammars which emphasize the differences between sets of sample strings rather than the regularities within a sample set.

On a more theoretical level, an investigation of further relationships between discriminant grammars and probabilistic automata would be interesting. For example, is it true that every cut-point event can be represented as a decision region of some discriminant grammar? It is also an interesting question whether a discriminant grammar can always be extended to accept arbitrarily noisy strings, that is, given a discriminant grammar D with characteristic grammar G and a cut-point θ, is it always possible to find a discriminant grammar D' with characteristic grammar G' and cut-point θ' such that $L^-(D,\theta) \subseteq L^-(D',\theta')$ and $L^+(D,\theta) \subseteq L^+(D',\theta')$ and $L(G') = V_T^*$?

It is not hard to imagine a host of other questions, such as the possibility of extending the sequential discriminative parsing scheme to grammars which are not LL(k) or the possibility of devising a discriminant grammar scheme based on ambiguous grammars. It is the authors hope that this thesis opens up a new area of study which will prove important in the future.

APPENDICES

APPENDIX A. LL(k) GRAMMARS AND TOP-DOWN PARSING

Of the three general classes of parsing algorithms (top-down, bottom-up, and tabular), the top-down method is of greatest relevance to this thesis. This technique allows us to reconstruct a LMCD in a forward (start symbol to sentence) manner, using information obtained from the input string to guide the parser in selecting productions. The parsing method described here is the LL(k) technique, which allows us to parse a certain class of unambiguous context-free languages in linear time. This class of languages is defined as follows:

> Let $G = (V_N, V_T, \Pi, S)$ be a context-free grammar.
>
> Let k be a natural number, and let α be a string over $V_N \cup V_T$. Then define

$$FIRST_k(\alpha) = \{x \in V_T^* \mid \text{either } |x| < k \text{ and } \alpha \overset{*}{\Rightarrow} x \text{ or } |x| = k \text{ and } \alpha \overset{*}{\Rightarrow} xy \text{ for some } y \text{ in } V_T^*\}$$

Informally, $FIRST_k(\alpha)$ is the set of k-symbol terminal prefixes of strings derivable from α. In the following, let the notation $\underset{lm}{\Rightarrow}$ stand for left-most derivation. Then G is LL(k) if and only if the three conditions:

1) $S \overset{*}{\underset{lm}{\Rightarrow}} xA\alpha \underset{lm}{\Rightarrow} x\beta\alpha \Rightarrow xv$

2) $S \overset{*}{\underset{lm}{\Rightarrow}} xA\alpha \underset{lm}{\Rightarrow} x\gamma\alpha \Rightarrow xw$

3) $FIRST_k(v) = FIRST_k(w)$

imply that $\beta = \gamma$.

LL(k) languages lend themselves in a natural way to deterministic top-down parsing. Informally, as we are reconstructing the derivation of a sentence z in L(G) and wish to know what production rule to apply to the non-terminal A in the sentential form $xA\alpha$, the decision can be determined by looking at the k terminals following the prefix x of z. In practice, an LL(k) parser may easily be programmed using either table look-up or recursive descent. A complete discussion of LL(k) techinques may be found in Aho and Ullman(1).

It is known that for any k the class of LL(k) grammars forms a proper subset of the unambiguous context-free grammars and furthermore that the class of LL(m) grammars properly includes the class of LL(n) grammars if m exceeds n. Given any language L, it is undecidable if L is generated by an LL(k) grammar, but the class of languages parsable by LL(k) methods is large enough, for example, to allow ALGOL to be thus compiled (See Lewis and Rosenkrantz(16)). An LL(1) grammar is called <u>simple</u> if and only if for each pair (A,a), where $A \in V_N$ and $a \in V_T$, there is at most one production of the form $A \rightarrow a\alpha$. Many of the

examples used in this thesis are simple LL(1) grammars.

APPENDIX B. STOCHASTIC GRAMMARS

In may situations, particularly in pattern classification problems, it is desirable to have a probability assignment over a language, that is, a mapping P: L→[0,1] such that

$$\sum_{x \in L} P(x) = 1$$

Considered as a set of ordered pairs, this function is called a stochastic language. Since interesting languages are usually infinite, it behooves us to specify an algorithm for the computation of this function in order to define it. One way to do this is to combine the probability computation with the language generation by means of a "stochastic grammar". See, for example, Booth and Thompson(2).

A stochastic grammar is a quintuple

$$F = (V_N, V_T, \Pi, S, P)$$

where V_N, V_T, Π, and S are the non-terminal set, the terminal set, the production set, and the start symbol, respectively, and P is a mapping from Π to the unit interval [0,1]. The characteristic grammar CHAR(F) is the ordinary grammar formed from the first four components of

F. The probability of a given derivation $\pi_1,\pi_2,\pi_3,\ldots,\pi_n$ is defined to be equal to the product

$$P(\pi_1)P(\pi_2)P(\pi_3)\ldots P(\pi_n)$$

Implicit in this definition is the following extremely important principle:

UNRESTRICTEDNESS ASSUMPTION: The probability of the application of any production depends only upon the presence of a given premise in a derivation and not upon how the premise was generated.

For all $x \in L(CHAR(F))$, the probability of x (denoted $\Gamma(x)$) is defined to be the sum of the probabilities of all distinct LMCD's of x. Note that if the grammar is unambiguous, there is only one term in this sum.

If the relation

$$\sum_{x \in L(CHAR(F))} \Gamma(x) = 1$$

is satisfied, then Γ is a true probability mass function and F is said to be consistent. In this case Γ is said to be the stochastic language generated by F ($\Gamma = L(F)$).

For each $A \in V_N$, let $\eta(A)$ be the subset of Π consisting of all productions whose premise is A. Then F is said to be proper if, for all $A \in V_N$,

$$\sum_{\pi_i \ \varepsilon \ \eta(A)} P(\pi_i) = 1$$

All stochastic grammars used in this thesis will be unrestricted, consistent, and proper.

BIBLIOGRAPHY

BIBLIOGRAPHY

(1) Aho, A.V. and Ullman, J.D. The Theory of Parsing, Translating, and Compiling. Prentice-Hall, Englewood Cliffs, N.J. 1972.

(2) Booth, T.L. and Thompson, A. "Applying Probability Measures to Abstract Languages." IEEE Transactions on Computers C-22 (May 1973) 442-449.

(3) Feller, W. An Introduction to Probability Theory and Its Applications, v. 1. Wiley, New York, 1968.

(4) Duda, R.O. and Hart, P.E. Pattern Classification and Scene Analysis. Wiley, New York, 1973.

(5) Freeman, H. "On the Encoding of Arbitrary Geometric Configurations." IEEE Transactions on Electronic Computers EC-10 (1961) 260-268.

(6) Fu, K.S. Sequential Methods in Pattern Recognition and Machine Learning. Academic Press, New York, 1968.

(7) Fu, K.S. Syntactic Methods in Pattern Recognition. Academic Press, New York, 1974.

(8) Fu, K.S. and Booth, T.L. "Grammatical Inference: Introduction and Survey -- Part I." IEEE Transactions on Systems, Man, and Cybernetics SMC-5 (January 1975) 95-111.

(9) Good, I.J. Probability and the Weighing of Evidence. Charles Griffin, London, 1950.

(10) Hopcroft, J.E. and Ullman, J.D. Formal Languages and Their Relation to Automata. Addison-Wesley, Reading, Mass., 1969.

(11) Kadota, T.T. and Sheep, L.A. "On the Best Finite Set of Linear Observables for Discriminating Two Gaussian Signals." IEEE Transactions on Information Theory IT-13 (1967) 278-284.

(12) Kanal, L.N. (ed.) Pattern Recognition. Thompson Book Co., Washington, D.C., 1968.

(13) Kirsch, R.A. "Computer Interpretation of English Text and Patterns." IEEE Transactions on Electronic Computers EC-13 (1964) 363-376.

(14) Kullback, S. Information Theory and Statistics. Wiley, New York, 1959.

(15) Lee, H.C. and Fu, K.S. Stochastic Languages For Picture Recognition TR-EE 72-17. Purdue University, Lafayette, Indiana, June 1972.

(16) Lewis, P.M. and Rosenkrantz, D.J. "An Algol Compiler Designed Using Automata Theory" Proceedings Polytechnic Institute of Brooklyn Symposium on Computers and Automata (1971).

(17) Mendel, J.M. and Fu, K.S. Adaptive, Learning and Pattern Recognition Systems: Theory and Practice. New York, Academic Press, 1970.

(18) Minsky, M. "Steps Toward Artificial Intelligence." Proceedings IRE 49 (1961) 8-30.

(19) Paz, A. Introduction to Probabilistic Automata. Academic Press, New York, 1971.

(20) Wald, A. Sequential Analysis. Wiley, New York, 1947.