COMPARISON OF PARSING TECHNIQUES FOR THE SYNTACTIC PATTERN RECOGNITION OF SIMPLE SHAPES

T.Bellone, E. Borgogno, G. Comoglio

DIGET, Politecnico, Corso Duca degli Abruzzi, 24, Torino, Italy
tamara.bellone, enrico.borgogno, giuliano.comoglio@polito.it

Commission III, WG III/8

KEY WORDS: algorithms, vision, pattern, recognition, understanding

ABSTRACT:

Syntactic Pattern Recognition is a procedure, widely used in Cartography and Remote Sensing, that trusts upon matching of sections of maps and/or images or 3D models with archetypes or objects (parsers). Parsing is a topic proper of Linguistics that can be considered as a basic step (syntax analysis) of the Syntactic Pattern Recognition procedure.
Considering a possible application of such technique to the automatic interpretation of imaged shapes, preliminary tests have been carried out onto simple geometric forms. An appropriate test image showing different geometric shapes has therefore been created. Parsing techniques have been applied as decision modules of the whole recognition path which is completed by some preliminary image processing steps. A number of algorithms are available for Parsing, for the needs of specific grammars: although not suited for any grammars, tabular methods help save time, as the Kasami method, remarkably simple to use: it works well in the case of context-free grammars, as reduced to the so- called Chomsky's normal form.
Languages used to describe noisy and distorted patterns are often ambiguous: one string or pattern can be generated by more than one language, so patterns belonging to different classes may have the same description, but with different probabilities of occurrence. Different approaches have been proposed: when a noisy pattern has two or more structural descriptions, it is proper to use stochastic grammars.
For the above said test it has been used a normal context free grammar over simple figures, that is a well designed specimen.
We also test a badly designed specimen using from the start a stochastic finite state grammar, which can be assimilated to a finite state Markov process: a final comparison of the results shall try to show the differences between those approaches.

1. INTRODUCTION

Language is based upon blending of discrete parts (phonemes, morphemes): we may suppose what really happens as one speaks, that is a passage from a starting point to a universe in progress of more and more complicated structures, and this process may be thought of as unique. However, as Minsky says, the same kind of process takes place when we try to understand a visual experience.
The meaning of a sentence relies upon the single words and upon their position. A sequence of words is seen as grammatically correct only when words appear in the framework of a certain scheme, however bias between sense and non-sense is only partially grammatical (grammar does not entirely control language).
Linguistic assemblages are shaped upon forms and rules. For a number of scholars, linguistic forms, at least the most common types, arise less from a sort of inner device proper to the language, than from the very way one thinks: in other words the special way our brain works must be taken into account. So vision and speaking are both based upon principles not quite different.
This is why some present procedures of Geomatics may be referred to logical and symbolic structures proper for Mathematical Logics and for Linguistics. Some improvements in GIS, Digital Photogrammetry and Remote Sensing are referred to as Computer Vision, Image Processing, Machine Learning, which are also linked to developments of Artificial Intelligence, which in turn is based upon Logics, Mathematical logics, Linguistics and Mathematical Linguistics.
An easy case of this cultural melting is Pattern Recognition, as used in the framework of Image Processing: it is a procedure, widely used in Cartography and Remote Sensing, that trusts upon matching of sections of maps and/or images or 3D-models with archetypes or objects (parsers). Also, parsing is a topic proper of Linguistics, which has been borrowed from cognitive sciences.
Syntactic Pattern Recognition consists of three major steps:

- preprocessing, which improves the quality of an image, e.g. filtering, enhancement, etc.
- pattern representation, which segments the picture and assigns the segments to the parts in the model
- syntax analysis, which recognizes the picture according to the syntactic model: once a grammar has been defined, some type of recognition device is required, the application of such a recognizer is called Parsing.

The decision whether or not the representation belongs to the class of patterns described by the given grammar or syntax (is syntactically correct) is made by a "parser".
Parsing is then the syntax analysis: this is an analogy between the hierarchical (treelike) structure of patterns and the syntax of language.
Patterns are built up by sub-patterns in various ways of composition, just sentences are built up by words and sub-patterns are built up by concatenating primitives (features) just words by concatenating characters.
Also, a texture is considered to be defined by subpatterns that occur repeteadly according to a set of rules.

A number of pattern recognition applications shows some distortion (due to noise). Also, patterns of a certain class are more frequent than others inside the same class. Quite the same as for natural languages, ambiguity should be kept into account: a sentence may actually bear different meanings, due to possible differences at recognition or interpretative level.

2. PARSING STRATEGIES FOR PATTERN RECOGNITION

2.1 The Theory of Formal Languages

A statistic approach to language started as the scholars realized the value of statistical methods in the recognition of speech. Hidden Markov's model was first used by the Author for modelling the language of "Evgenij Onegin" (Markov, 1913).
In the theory of formal languages, a language is defined as a set of strings: a string is a finite sequence of symbols chosen from some finite vocabulary. In natural languages, a string is a sentence, and the sentences are sequences of words chosen from some vocabulary of possible words. A grammar is defined as a four-tuple:

$$\mathbf{G = (N, T, P, S)}$$

where N and T are the non terminal and terminal vocabularies of G, P is the set of production rules, and S is the start symbol.
A formal language is indeed defined by:

- A terminal vocabulary of symbols (the words of the natural language)
- A non terminal vocabulary of symbols (the syntactic categories, e.g. noun, verb)
- A set of productions (the phrase structure rules of the language)
- The so called start symbol

We start from a special non terminal S, and S is replaced by the string on the right side of a chosen production rule. The process of rewriting a substring according to one of the rewriting production rules continues until the string consists only of terminal symbols:

$$S \rightarrow aS \mid bS \mid \varepsilon$$

where the symbol | indicates "or" and ε is the null string. The succession of strings that result from the process is a derivation from the grammar: to find a derivation (a parse) for a given sentence (sequence) is called parsing.
As to the latter approach, let's remind that in the Theory of Formal languages, Chomsky divides languages in classes, thus forming a hierarchy of languages, based upon different grammars:

- Unrestricted grammars (0-type grammars)
- Context-sensitive grammars (1-type grammars)
- Context-free grammars (2-type grammars)
- Regular grammars or finite state grammars (3-type grammars)

The most general grammar is obviously the 0-type, which bears no limits for rewriting rules: for the other types, such restrictions are regularly increasing. Types 0 and 1 are able to describe natural languages as the other two, much simpler to manage from a computational viewpoint, are more suited into limited backgrounds and have been hitherto used for artificial languages.
In the 1-type grammar, rewriting rules restraints bear that the right side of a rule should have at least as many symbols as the left side.
For the 2-type grammar, all rules should have just one non-terminal symbol on the left side
For 3-type grammar, the right side has only one terminal symbol, or one terminal symbol and a non-terminal one for every production rule.
An attraction of the use of a syntactic method for Pattern Recognition is the possibility of description and recognition of an object, a scene, a texture: but noise may complicate the process of computing the string structure: extraneous primitives may be generated by spurious edges, or actual primitives of some shape may be undetected due to the poor quality of the image.
Another problem is the ambiguity of the primitives that are extracted to represent the patterns, how this ambiguity can be accounted for in the model or in the analysis (in the construction of the grammar or in the parsing).
A grammar normally divides strings in just two classes: the grammatically correct ones, and the others. In any case, ambiguity is very frequent, and it is even deliberately pursued as sometimes happens in poetry.
The following different approaches have been proposed:

- approximation
- transformational grammars
- stochastic grammars
- similarity and error-correcting parsing

The use of approximation reduces the effect of noise and distortion at the preprocessing and primitive extraction stage.
The second approach defines the relations between the noisy pattern and the corresponding noise-free pattern by a transformational grammar.
Many efforts have been devoted to construct more sophisticated grammars, like stochastic and fuzzy grammars. Although context free grammars or transformational grammars can represent the phrase structure of a language, they tell nothing about the relative frequency or likelihood of a given sentence. It is usual in context free grammar to use recursive productions to represent repetition, however one can generate sentences which are technically grammatical, but not always acceptable. Stochastic approach supplies a solution to this problem: each production in a stochastic grammar is assigned a probability of selection, a number between zero and one. During the derivation process, rules are selected for rewriting according to their assigned probabilities. Each string has a probability of occurrence computed as the product of the probabilities of the rules in its derivation.
When a string has two or more parses, we can use the more probable parse as a description of the string (pattern): the most probable parse is the one according to which the generated string has the highest probability.
However, what we already know about probable occurrence plays a meaningful role. The parsers are made up according to a likelihood criterion. However, parsers may also be built according to a further criterion, i.e. the Bayes's theorem.
In this case, some utter a priori information is required about the starting probability to deal with one class of patterns or another one.
When a distorted pattern cannot be accepted by any grammar, an error-correcting parsing, based upon a similarity criterion, can be used: a way to handle noisy patterns is the use of similarity measures instead of stochastic grammars: a similarity or distance measure is defined between a sentence representing a known pattern and sentence representing an unknown pattern.

The distance between two strings is defined in terms of number of errors (insertion, deletion, substitution.).

2.2 Parsing algorithms

In accordance with the needs of different grammars, a certain number of Parsing algorithms are in use.
Parsing procedures may be top-down or bottom-up type, as one starts from initial symbol S, operating substitutions until only terminal symbols are present, such as to fit the clause, or as one starts from the string backward till the start symbol S.
Besides, the language classes as arranged by Chomsky are related to a number of reconnaissance devices (automata):
0-type languages: Turing machines
1-type languages: bounded automata
2-type languages: pushdown automata
3-type languages: finite state automata.
Chart algorithms are of special interest, because they are simple, although they are referred to context-free grammars.
In the following, context-free as well as stochastic regular grammars have been used. So, we would recall a typical chart algorithm, the Kasami's one, and also remind of equivalence between stochastic regular grammars and Hidden Markov Models (HMMs).
Kasami's algorithm is suited to context-free grammars, as they are transformed in the so called Chomsky normal form (CNF).
To get CNF, we convert all rules to such production rules that all non terminal symbols would yield either two other non terminals or one terminal:
$A \rightarrow a$
$A \rightarrow BC$
Be w= $a_1a_2 \dots a_n$ a string whose pertinence to a given gramamr is to be tested, the grammar being already reduced to the CNF.
The algorithm is basically a triangular parsing table, whose elements are t_{ij} for $1 \leq i \leq n$ e $1 \leq j \leq n-i+1$. Every t_{ij} should have a value being a sub-set of N. The non terminal symbol shall be inside t_{ij} if, and only if:

$A \rightarrow a_1 a_{i+1} \dots a_{i+j-1}$..

The table is assembled as follows:
- one states t_{i1}=A if $A \rightarrow a_i$
- one states t_{ij} =A even for a single k, such that $1 \leq k < j$ if $A \rightarrow BC$ is to be found in P, having B present in t_{ik} and C in $t_{i+k,j-k}$.

The string shall belong to the said language just in case S shall be found into t_{1n}
Stochastic regular grammars are equivalent to Hidden Markov Models.
A Hidden Markov Model has, properly, hidden states: so, just a series of symbols is present, which allows a probabilistic inference towards the related sequence of states.
A Hidden Markov Model (HMM) is specified by a set of parameters:
- the set of states $S = (s_1, s_2, \dots, s_N)$
- state sequence $Q = (q_1, q_2, \dots, q_k)$
- the prior probabilities (π) are the probability distributions of q_i being the first state of a state sequence
- the transition probabilities (a_{ij}) are the probability to go from a state i to a state j, i.e P ($q_j| q_i$)
- the emission probabilities (e) are the probability of observing x when the system is in the state q_i $p(x|q_i)$

One can calculate:
- the likelihood of a sequence of observations with respect to a HMM by means of the so called "forward algorithm"
- the most probable sequence of corresponding states (most probable path) by Viterbi algorithm
- the HMM parameter estimation by forward-backward algorithm

Let $f_k(i) = \Pr(x_1 \dots x_i, \pi_i = k)$ be the probability to observe the sequence $x= x_1 \dots x_i$, at the same time, to be in the state k. The Forward algorithm allows recursive calculation of x probability to be done. The steps of algorithm are as follows:

- initialisation: $f_0(i) = \pi_i\, e(x_i)$
- recursive step: $f_i(i) = e_l(x_i)\Sigma_k (f_k(i-1)a_{kl})$
- termination: $\Pr(x)=)\Sigma_k (f_k(n)a_{kl})$

Viterbi's algorithm, in the turn, lets associate to a sequence, the related most probable asset of otherwise hidden states: computation is quite analogous, just substituting in the Forward algorithm table the sum with the maximum search, according to the process: probability at each case corresponding to the Forward algorithm comes from the sum of some terms; however, for Viterbi's algorithm, only the maximum one of abovesaid terms is used.
Viterbi's algorithm keeps, fore evrey state and for every position, a pointer to the preceeding state, so as to trace backwards the most probable path.
The following HMM has the alphabet: 0,1; a_{11} ,a_{12} , a_{13} ,a_{21} ,a_{22} a_{23} , a_{31} , a_{32} , a_{33} are the transition probabilities and parameters $e_1(0)$, $e_2(0)$, $e_3(0)$, $e_1(1)$, $e_2(1)$, $e_3(1)$ represent the emission probabilities.
The corresponding stochastic regular grammar is:

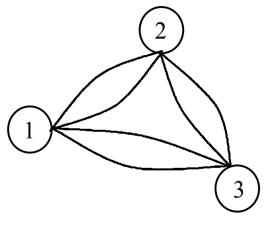
$S \rightarrow X_1$

$$X_1 \rightarrow \underset{p_{11}}{0X_1} \mid \underset{p_{12}}{1X_1} \mid \underset{p_{13}}{0X_2} \mid \underset{p_{14}}{1X_2} \mid \underset{p_{15}}{0X_3} \mid \underset{p_{16}}{1X_3}$$

$$X_2 \rightarrow \underset{P_{21}}{0X_1} \mid \underset{p_{22}}{1X_1} \mid \underset{p_{23}}{0X_2} \mid \underset{p_{24}}{1X_2} \mid \underset{p_{25}}{0X_3} \mid \underset{p_{26}}{1X_3}$$

$$X_3 \rightarrow \underset{P_{31}}{0X_1} \mid \underset{p_{32}}{1X_1} \mid \underset{p_{33}}{0X_2} \mid \underset{p_{34}}{1X_2} \mid \underset{p_{35}}{0X_3} \mid \underset{p_{36}}{1X_3}$$

$p_{11} + p_{12} = a_{11}$
$p_{13} + p_{14} = a_{12}$
$p_{15} + p_{16} = a_{13}$
$p_{11} / (p_{11} + p_{12}) = e_{11}(0)$
$p_{13} / (p_{13} + p_{14}) = e_{12}(0)$
$p_{15} / (p_{15}+ p_{16}) = e_{13}(0)$



$e_1(0) = e_{11}(0) + e_{12}(0) + e_{13}(0)$

We repeat for the p_{2j} and p_{3j} probabilities.
We can obtain the set of probabilities of the production rules of the stochastic regular grammar, given the set of emission probabilities and the transition matrix of HMM, and viceversa.
Parsing is finding an optimal derivation for a given sequence (string): it can be tought as an alignment of non terminal symbols (hidden states) of the grammar and terminal symbols of the sequence (string), just as Viterbi's alignement of a sequence positions to HMM states.
Moreover, the theory of stochastic automata define the class of languages accepted by stochastic automata.

A stochastic finite state automaton is a five tuple SA= (Σ, Q, M, π_0 , F) where Σ is a finite set of input symbols for the strings (the alphabet) , Q is a finite set of internal states, M is a mapping of Σ into the set of n$\times$n stochastic state transition matrices, π_0 is the n-dimensional initial state distribution vector and F is a finite set of final states.
Indeed the generation process from a stochastic finite state grammar can be assimilated to a finite state Markov process.
Let $G_S = (V_N , V_T, P_S , S)$ be a stochastic grammar, we can construct a stochastic automaton SA = (Σ, Q, M, π_0), accepting languages generated by G_S : T(SA)
The alphabet Σ is equal to the set of terminal symbols V_T ; the state set Q is the union of the set of non terminals V_N and the states T and R, state of termination and of rejection respectively; π_0 is a row vector with the component equal to 1 in the position of state S, the other components equal to 0; the state transition matrices M are formed on the basis of the stochastic productions; finally a n vector π_f represents the final state.
Let's consider the stochastic finite state grammar $G = (V_N , V_T, P_S , S)$, where:
V_N = (S, A), V_T = (a, b) and P_S :

$S \rightarrow aA \quad p_1= 1$
$A \rightarrow aA \quad p_2 = 0.8$
$A \rightarrow b \quad p_3 = 0.2$

We have, according to the above described procedure: Σ = (a, b), Q = (S, A, T, R), π_0 = (1 0 0 0), and F = T.
We can construct the transition state matrices M(a) and M(b):

$$M(a)=\begin{array}{c|cccc} & S & A & T & R \\ \hline S & 0 & 1 & 0 & 0 \\ A & 0 & 0.8 & 0 & 0.2 \\ T & 0 & 0 & 0 & 1 \\ R & 0 & 0 & 0 & 1 \end{array} \qquad M(b)=\begin{array}{c|cccc} & S & A & T & R \\ \hline S & 0 & 0 & 0 & 1 \\ A & 0 & 0 & 0.2 & 0.8 \\ T & 0 & 0 & 0 & 1 \\ R & 0 & 0 & 0 & 1 \end{array}$$

According to the Markov chains theory, we can calculate for example the probability of the string x = aab.
$M (aab) = \pi_0 M^2 (a) M (b) \pi_f$

$$M(aab)=\begin{vmatrix}1 & 0 & 0 & 0\end{vmatrix}\begin{vmatrix}0 & 0.8 & 0 & 0.2\\ 0 & 0.64 & 0 & 0.36\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\end{vmatrix}\begin{vmatrix}0 & 0 & 0 & 1\\ 0 & 0 & 0.2 & 0.8\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1\end{vmatrix}\begin{vmatrix}0\\ 0\\ 1\\ 0\end{vmatrix}=0.16$$

If we calculate the probability of the string by means of forward algorithm of HMMS, we obtain the same result.
Indeed, we can consider the hidden states: A and T and the following parameters:

$\pi(A) = 1, \pi(T) = 0;$
$e_A(a) = 1, e_T(b) = 1;$
$a_{AA} = 0.8, a_{AT} = 0.2, a_{TT} = 0.8;$
$f_A (a) = \pi(A)\, e_A(a) = 1;$
$f_T (a) = \pi(T)\, e_T(a) =0;$
$f_A (a,a) = 0.8, f_T (a,a) = 0;$
$f_A (a,a,b) = 0, f_T (a,a,b) = 0.16;$
$f (a,a,b) = f_A (a,a,b) + f_T (a,a,b) = 0.16.$

If we calculate the probability of the string x= aab taking into account how it is generated:

$$S \rightarrow aA \rightarrow aaA \rightarrow aab$$

we obtain the same result:

$$p(x) = p(aab) = 1\times 0.8 \times 0.2 = 0.16.$$

2.3 Application test

We mean to test a badly designed specimen either by proper use of a normal grammar after a pre-treatment of the specimen, or using from the start a stochastic grammar.
First, in order to evaluate how well such approach could be applied to the automatic interpretation of images preliminary tests have been carried onto simple geometric images.
It is here shown an example addressed to draft a possible operational path for Parsing based pattern recognition of simple geometric entities.

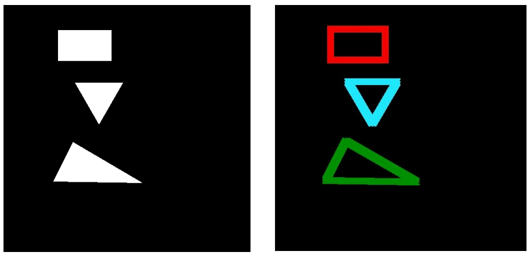


Fig. 1 Test image and polygon grouped pixels image.

An appropriate test image has been created showing three different geometric figures: a rectangle, a scalene triangle and a equilateral triangle. The goal is to verify if the implemented grammar could correctly decide if one figure is or not an equilateral triangle.
The recognition process goes on in the following way:

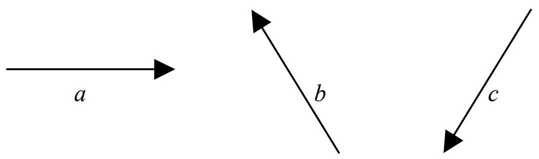
- a preliminary identification, based on radiometric/spectral discriminants, of the pixels of the image probably belonging to the searched objects is firstly carried out;
- the selected pixels are then grouped in different distinct geometric entities using neighbourhood and region growing criteria (different colors in the image below);
- for each entity a frame window is clipped from the original image and a Förstner filtering and thresholding algorithm is applied in order to select pixels most probably representing the vertices of the figure which has to be recognized;
- assuming that figures are closed polygons vertices coordinates are used to define length and direction of the connecting lines. These are the geometric primitives used in the parsing grammar. A simple translation from numbers to letters (defined when grammar has been defined) allow to transfer information to the parsing engine algorithm which has to decide if the object belongs or not to the defined grammar, that is if that polygon is or not an equilateral triangle.

All these steps have been implemented using the IDL programming language. We do not intended to deeply describe well known image processing algorithms. Otherwise, we care to briefly describe how the parsing algorithm works. It has firstly to be structured, defining the deciding grammar. This is a static part of the program; in fact, once defined, it never changes during the recognizing process. Changing grammar means to change the program text. In the future we intend to define a standard text file the user can fill off-line to define the different grammars he wants to use. Such file could be directly be read by the program while executing. Grammar has been structured as a matrix, with a column number equal to the number of the

generic values A_i, and with a row number equal to the maximum number of values (terminal and not) that each generic value A_i can assume. Each generic value A_i (first line) determines its own column with the possible values it can assume. Terminal values are listed in the last line. The matrix is a sparse one.
The program reads the string corresponding to the translation of the geometric primitives in characters and it automatically generates the Kasami table for that string. This table size obviously depends on the stringth length. Strings belong to the grammar if S can be found in the last row, first column of the Kasami table.
Used grammar is a context-free one and it is suitable for the recognition of different size equilateral triangles:

$S \rightarrow AA_1C$	$A_1 \rightarrow AA_2C$	$B_2 \rightarrow BB_1$
$A_1 \rightarrow b$	$A_2 \rightarrow aB_3C$	$B_1 \rightarrow b$
$A_1 \rightarrow aB_2C$	$B_3 \rightarrow bB_2$	$C \rightarrow c$

The terminal values a, b and c represent the following primitives :



Such grammar reduced to the Chomsky Normal Form can be defined as follows :

$S \rightarrow A_3A_4$	$A_4 \rightarrow A_1C$	$A_3 \rightarrow A$	
$A_1 \rightarrow A_3A_5$	$A_5 \rightarrow A_2C$	$B_2 \rightarrow B_4B_1$	
$B_4 \rightarrow b$	$A_1 \rightarrow b$	B_3	$A_2 \rightarrow A_3A_6$
$A_6 \rightarrow B_3C$	$B_1 \rightarrow b$	$A_1 \rightarrow A_3A_7$	$A_7 \rightarrow B_2C$
$B_3 \rightarrow B_4B_2$	$C \rightarrow C$		

It has been verified that the program can correctly label as 'equilateral triangles' the ones corresponding to the strings
w = "*abc*" and y = "*aabbcc*"; while it labels w = "*aba*" and
y = "aabbca" as 'not –equilateral triangles'.
Obviously, images are never perfect: they are regularly noise poisoned; thus we shall use a stochastic grammar. The chosen one is regular, as it is equivalent to a HMM, so that we can use the corresponding algorithms, which are a well known powerful tool.
An appropriate test image has been created showing a distorted version of an equilateral triangle

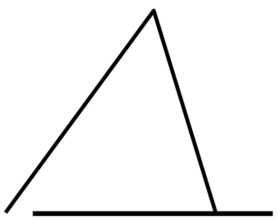


Fig. 2 Test image of a noisy pattern (equilateral triangle).

A stochastic regular grammar has been introduced in order to describe an equilateral triangle and some other distorted versions (Fig. 3):

$S \rightarrow aA_1$
$A_1 \rightarrow b_1A_2$, $A_1 \rightarrow b_2A_2$
$A_2 \rightarrow c_1$, $A_2 \rightarrow c_2$

The object has been subdivided in subpatterns, corresponding to the string $x = ab_2c_2$.
Let's suppose to know probability to get each string x:
$p(a\, b_1\, c_1) = 0.01$
$p(a\, b_2\, c_1) = 0.3$
$p(a\, b_1\, c_2) = 0.19$
$p(a\, b_2\, c_2) = 0.5$
From probability theory ensues that the probabilities of the production rules are:
$p_1 = 1$
$p_2 = 0.032$
$p_3 = 0.968$
$p_4 = 0.31$
$p_5 = 0.69$

The hidden states of the corresponding model are: A_1, A_2 and T.
The parameters of the corresponding HMM are:
$\pi(A_1) = 1$, $\pi(A_2) = 0$, $\pi(T) = 0$;
$e_{A1}(a) = 1$, $e_{A2}(b_1) = 0.032$, $e_{A2}(b_2) = 0.968$, $e_T(c_1) = 0.31$,
$e_T(c_2) = 0.69$;
$a_{A1A2} = 1$, $a_{A2T} = 1$

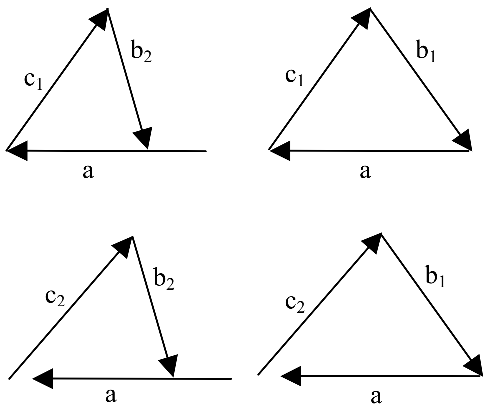


Fig. 3 An equilateral triangle and some distorted versions.

The "Forward" algorithm shall give us probability of a sequence of observations, for instance of the string $x = a\, b_2\, c_2$:
$f(a\, b_2\, c_2) = 0.67$.
Viterbi's algorithm, in the turn, lets associate to such a sequence, the related most probable asset of otherwise hidden states (the parse of the string):
$A_1\, A_2\, T$.

3. FORECASTS

The ab_2c_2 string of the previously described example, may belong also to the classe of square triangles.

In case of syntax analyzers (parsers) planned on the basis of context-free, non stochastic (i.e. traditional) grammars, the two classes of patterns could not be distinguished: that could mean the string belong to both patterns.
However, what we already know about probable occurrence plays a meaningful role.The parsers, in many cases, are made up according to a likelihood criterion. However, parsers may also be built according to a further criterion, i.e. the Bayes's theorem Stochastic context free grammars are more powerful in order to describe languages: additional rules allow to create nested, long-distance pairwise correlations between terminal symbols.
In the case of complex patterns, with more articulated relations between sub-patterns, regular grammars (even in stochastic form) are not adequate: stochastic context-free grammars, also more compact, should prove more efficient, altough more expensive in terms of memory and computational time.
A variety of Cocke-Younger-Kasami (CYK) algorithm allows to find out an optimal parse tree (alignment problem) for stochastic context free grammars in Chomsky Normal Form; the so called "inside algorithm" allows to find out probability of a given sequence (string) if the grammar is previously known.
The inside algorithm can be compared with the forward algorithm for HMMs, the same as the CYK algorithm can be compared with the Viterbi algorithm used for HMMs.

References

Aho, A., Ullman, J., 1972. *The theory of Parsing Translation, and Compiling*. Prentice Hall, Englewood Cliffs.

Chomsky, N., 1957. *Syntactic structures*. Mouton, The Hague

Fu, K., 1974. *Syntactic Methods in Pattern Recognition*. Academic Press, New York.

Markov, A., 1913. An example of statistical investigation in the text of "Eugene Onegin". In: *Proceedings of the Academy of Sciences of St. Petersburg.*

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In: *Proceedings of the IEEE,* 77 (2).

Rabiner, L., Juang, B. 1993. *Fundamentals of Speech Recognition*. Prentice Hall, NJ.

Sester, M., 1992. Automatic model acquisition by Learning. In: ***The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*,** Whashington USA, Vol. XXIX, Part B3.

Resch B. Hidden Markov Models.
http://www.igi.tugraz.at/lehre/CI