On the Procrustes Algorithm for Feature Extraction*

George Svetlichny†

Abstract

We present and treat here the Procrustes algorithm of E.R. Caianiello and others.[1] We give a general description of the algorithm and a few examples; further details can be found in the references. The algorithm serves as a focal point for a large class of general problems which are now under study. We present here the setting for these problems.

In all our purposeful activities we make use of various objects whether they be physical or conceptual. We use such objects because certain of their features suit our purpose and we focus our attention only on these features. Thus from the potential infinity of features or properties that every object can manifest we select only a few. It often happens however that our selection of features becomes inadequate for our purposes and we must come to see our objects in terms of different set of features. The procedure of arriving at such a new selection of features envolves two basic processes. The first one of these may be called fragmentation wherein the given object is analyzed in greater and greater detail. The given features are thus analyzed into their own constituent features and so on to any degree. The second process may be called that of merger in which several given features are recognized to be tightly bound together and so are coalesced into one. The fragmentation process is often considered to have a great element of arbitrariness to it, namely what is usually considered to be sufficient is to perform a fragmentation into parts so small that all the new useful features can be obtained by merger from them. Such a deceptively simple viewpoint hides a host of sins and a deep analysis of the fragmentation process is much overdue owing to its apparent simplicity when compared to that of the merger process. We only note this in passing for we too shall focus mainly on the merger process and make only occasional references to fragmentation, leaving for another place the study of problems relating to it.

Because of the general pervasiveness of these two processes any attempt at the mechanization of intelligence must undertake a study of the mechanization of these activities. Such a mechanization in its turn will lead to a better understanding of these processes thus providing new tools for analyzing them.

*Written at the Laboratòrio de Cibernètica del CNR, Arco Felice, Napoli, Italy in 1970.

†Departamento de Matemática, Pontifícia Universidade Católica, Rio de Janeiro, Brazil
svetlich@mat.puc-rio.br http://www.mat.puc-rio.br/~svetlich

[1]Unfortunately, all references have been lost.

It should be pointed out quite early that a successful analysis of a given object can be accomplished only by a suspension of certain a-priori expectations. We must look and notice the new features that are in fact present, regardless of what we may have a-priori conceived or expected to find. Our attitude here differs fundamentally from the usual approach of pattern analysis and recognition in which a given object must be analyzed in terms of a-priori given features or be placed into any one of a-priori given classes. We are here attempting to extract features from an object without a priori knowing what they should be. We are thus attempting to mechanize an act of intelligence on a higher level than that of recognition; namely that of noticing. Rather than recognizing something that we know we must notice something that is new.

Of course we can never eliminate all a-priori elements, we can only push them to a higher level. One can raise the following objection. The act of noticing can be considered as an act of recognition of a particular mental state, and therefore in the mechanization of the merger process all one is doing is merely combining the various small features, first in one way, then in another, until finally, we hit upon that collection of mergers in which the new collection of features stands in a particular a-priorily conceived relationship to the object under consideration. We are hence reduced to an act of recognition of an a-priori pattern. We rule that such an objection results from a confusion in levels of meaning similar to the one that reigned in logic before the distinction between theory and metatheory was made. A good linguistic methodology is needed to avoid such confusions but it is yet to be developed.

The a-priori element that remains in the process of merger is the principle by which features are merged to generate others. Such a principle must reflect the purpose for which we need the new features and can itself be subject to mechanical manipulation in a machine that automates the merger process.

In its most general form the merger principle can be stated as follows: any collection of features that are very tightly related either spatially, temporally, functionally, or in any other well perceived sense can become unified into a single feature and treated as such. That this operates in the human nervous system can be attested by innumerable works on the psychology of perception.

It is generally useful to consider that we have before us not a single object whose features we must merge in some way, but a whole class of objects. In this way part of the merger procedure can operate by comparing the different objects among themselves and so extracting features common to all members of the collection. This approach is necessary for instance in studying the temporal behavior of some object where it is not enough to have the object at some one instant of time, but it is necessary to study the relationships existing between the states of the object for a large number of time instances. We do not exclude the original case of a class containing a single object.

We assume therefore that we are presented with a class of similar objects, and furthermore that each member of this class has already been fragmented into small constituent features. The general Procrustes problem now is to collect together groups of small features that are closely tied together in some well described manner. These groups now form the new features that we are seeking.

Having accomplished this merger once we can contemplate performing it once more, collecting groups of new features together to form still newer features and so on to any degree. Thus we can establish a hierarchy of features in which each feature on a given level is obtained by merger from features at the next lower level and so on until we reach the lowest level which is that of the original fragmentation. There is a problem of optimization at each level in that the features that are formed must in some sense be the optimal ones for the description of the object on that level. This problem is closely tied to the use to which we put the hierarchies so obtained.

The successful automation of the Procrustes program for any class of situations is extremely useful for such an automation provides in a very real sense a measuring instrument. Most objects that we deal with already have a hierarchy of features that has been constructed historically by our dealings with them. Next to such a natural hierarchy we can then place one that has been constructed by well prescribed procedures using this latter one as a standard. Comparing the two hierarchies leads to a deeper understanding of both the objects involved and of the process of hierarchy formation. A vast class of problems is thus opened up for study.

We now describe an algorithm for hierarchy formation for a particular class of objects, namely the class of texts written in some given language. By a text we shall mean a string of letters $\lambda_1, \lambda_2, \ldots, \lambda_n$ taken from some finite alphabet Λ^0 and which satisfies all the rules of formation existing in the language. The language can be identified with the set of these rules of formation or alternatively with the set of all texts, that is with the set of all allowable strings of letters. The problem now is to recover the structure of the language knowing a-priori only the sequences of letters making up a certain subclass of the texts.

We shall assume that the texts have a given hierarchical structure in that the sequences of letters making up the text can be viewed at various levels. The lowest level is to view the text just as the sequence of letters and this corresponds to the original fragmentation of the object into some starting set of features, namely the individual letters. On the next higher level certain consecutive sequences of letters occurring in the text are to be treated as units called syllables so that the original text can also be written as a sequence of syllables $\sigma_1\sigma_2\cdots\sigma_n$ where each σ_i stands for a sequence of letters of Λ^0. The set of syllabic symbols forms a new finite alphabet which we call Λ^1; each letter of this alphabet is coded in terms of the letters of Λ^0. To express this latter fact we define for any alphabet A the set A^* of all possible finite strings, including the empty string, of letters of A. We thus have $\Lambda^{1*} \subset \Lambda^{0*}$

Proceeding again with the text written in terms of syllables we can pass on by this process of "syllabification" to the next level in the hierarchy to obtain a new finite alphabet Λ^2 which is coded in terms of the letters of Λ^1. We envisage this procedure continuing until we reach some highest level Λ^k. We thus assume that the language is provided with a hierarchy of finite alphabets $\Lambda : \Lambda^{k*} \subset \Lambda^{k-1*} \subset \cdots \subset \Lambda^{1*} \subset \Lambda^{0*}$ in which each alphabet is coded in terms of the next lower one. Every text of the language we assume can be written in any of the above alphabets.

The Λ hierarchy must correspond to some definite structure of the language and not be just arbitrary. Concerning this point we shall only make the following general assumptions. We want that the rules of formation that exist in the language become less rigid as we move higher in the hierarchy of alphabets. Thus at the lowest level there should be strong restrictions on what short sequences of letters are allowed, at the next higher level there should be fewer restrictions, and at the highest level practically any sequence of symbols from Λ^k should correspond to an acceptable text. We shall not here endeavor to make the notion of such progressively freer hierarchies more precise because on the basis of this general notion we shall presently state an algorithm that constructs another hierarchy of alphabets $\Sigma : \Sigma^{H*} \subset \Sigma^{H-1*} \subset \cdots \subset \Sigma^{1*} \subset \Sigma^{0*} = \Lambda^{0*}$. It is only after a certain amount of experimentation on the relationship of the two hierarchies that a more precise notion will be formulated. Barring the existence of certain rules of formation and barring a poor choice of Λ we can extract the following heuristic principle from the idea of a progressively freer hierarchy. If at a given level a string of symbols σ can be followed by very many symbols as compared to some other string τ which can be followed by relatively few, then it is a good guess that the string σ ends at the same place as a higher level functional unit while the string τ must always end somewhere in the interior of such a unit. This is because on a higher level the rules of formation are freer and thus allow for a greater number of possibilities to follow certain occurrences of σ while the possibilities for following τ are governed by the rules on a lower level. To incorporate this principle into an algorithm for constructing Σ^{i+1} from Σ^i it is convenient to introduce the following function defined on strings of symbols. If $\sigma_1 \ldots \sigma_n$ is a string of letters of Σ^i then $h(\sigma_1 \ldots \sigma_n)$ is defined as the number of different symbols of Σ^i that are found following the string in a given text or set of texts. If a string does not occur at all or stands only at the very end of some text then $h = 0$, otherwise $h > 0$. If the function h is not constant on Σ^i then Σ^{i+1} can be described as follows:

1. We first choose a number $h_i^0 > 0$.

2. We let any string $\sigma_1 \ldots \sigma_n$ become an element of Σ^{i+1} if and only if $h(\sigma_1 \ldots \sigma_n) \geq h_i^0$ and no prefix satisfies this property.

This criterion clearly corresponds to the above heuristic principle, for $h \geq h^0$ is a signal that the string terminates a higher order unit, while forbidding this relation to every prefix means that this higher order unit is not properly contained within the string; that is, that it is not the complementary suffix.

The above criterion also allows for a step by step algorithm for constructing Σ^{i+1}. This is the Procrustes algorithm. The algorithm starts with Σ^i and then iterates the following step: at each step we place the strings with $h \geq h_i^0$ into Σ^{i+1} and increase the remaining strings in all possible ways by one letter.

Repeatedly applying this algorithm H times we obtain a hierarchy of alphabets $\Sigma^{H*} \subset \Sigma^{H-1*} \subset \cdots \subset \Sigma^{1*} \subset \Sigma^{0*}$. Each code $\Sigma^{i+1} \subset \Sigma^i$ has the following characteristic property. If $s \in \Sigma^{i+1}$ and s' is a suffix of s then either $s' \in \Sigma^{i+1}$ or it has a prefix in Σ^{i+1} but never both. This follows immediately from the

fact that $h(s') \geq h(s) \geq h_0$ and thus by our criterion either $s' \in \Sigma^{i+1}$ or the same for a prefix but not both. Codes having this property are called closed instantaneous codes (CIC).

Concerning the convergence of the algorithm we note the following. If h_0^i is no greater than the minimum of h on Σ^i then $\Sigma^i = \Sigma^{i+1}$. If h_0^i is greater than the maximum of h on Σ^i then either the algorithm diverges in that some strings grow forever (which can happen if an infinite number of texts are presented) or else each string grows until $h = 0$ and disappears at the next step and $\Sigma^{i+1} = \emptyset$. Somewhere in between an optimal situation is obtained. If h is constant on Σ^i then we must either terminate our hierarchy at this point or seek new ways of extending it. Various criteria have been constructed for deciding what to do in such a case.

One aspect of the algorithm should be commented upon, and that is that its operation suppresses any information contained in relative frequencies of occurrence of the various symbols at each level. Thus if σ_1 is only followed by σ_2 and σ_3 and if $\sigma_1\sigma_2$ occurs a thousand times in the text $\sigma_1\sigma_3$ occurs only ten times then $h(\sigma_1) = 2$ and the ten occurrences of $\sigma_1\sigma_3$ count, on this level, as much as the thousand occurrences of $\sigma_1\sigma_2$. The reason for this attitude is that texts are normally constructed in terms of higher level units and the relative frequency of occurrence should be determined by the use made of those. On the other hand, certain occurrences are indeed rare or constitute errors and should be considered as exceptions. If these situations are present in the text the operation of the algorithm can be distorted and the convergence spoiled. We have been able to eliminate such undesirable situations by finding various criteria for testing whether the convergence is satisfactory.

We can now pass on to some concrete examples. Let us take the CIC given by:

$$\Lambda^1 = \begin{matrix} 1 & 31 & 322 & 32312 & 3231321 \\ 2 & 321 & 32311 & 323131 & 3231322 \end{matrix}$$

where of course $\Lambda^0 = \{1, 2, 3\}$. We take Λ^{1*} for the set of texts. Taking $h_0 = 3$ we arrive at $\Sigma^1 = \{1, 2, 31, 32\}$; starting now with Σ^1 and taking $h_0 = 4$ we arrive at $\Sigma^2 = \{1, 2, 31, 321, 322, 3231\}$ and starting again with $h_0 = 6$ we arrive finally at $\Sigma^3 = \Lambda^1$. The above illustrates the general fact that if $\Lambda^{1*} \subset \Lambda^{0*}$ is a CIC and the set of texts is Λ^{1*} then the highest level of the Σ hierarchy coincides with Λ^1. The intermediate levels reflect the detailed internal structure of the code.

As another example let us consider the Italian alphabet written in Morse code. Let us first add the symbol $\otimes$ to denote blank and consider the following strings which constitute Λ^1:

A	$\cdot - \otimes$	H	$\cdot\cdot\cdot\cdot \otimes$	Q	$- - \cdot - \otimes$
B	$- \cdot\cdot\cdot \otimes$	I	$\cdot\cdot \otimes$	R	$\cdot - \cdot \otimes$
C	$- \cdot - \cdot \otimes$	L	$\cdot - \cdot\cdot \otimes$	S	$\cdot\cdot\cdot \otimes$
D	$- \cdot\cdot \otimes$	M	$- - \otimes$	T	$- \otimes$
E	$\cdot \otimes$	N	$- \cdot \otimes$	U	$\cdot\cdot - \otimes$
F	$\cdot\cdot - \cdot \otimes$	O	$- - - \otimes$	V	$\cdot\cdot\cdot - \otimes$
G	$- - \cdot \otimes$	P	$\cdot - - \cdot \otimes$	Z	$- - \cdot\cdot \otimes$

For the texts we take any string of Λ^1 with the proviso that it does not begin with $\otimes$, that it must end in a single $\otimes$, and that no more than two $\otimes$ be allowed in a row. The first difficulty with this situation is that h is constant on Σ^0. If we look at h on strings of two symbols then it is still constant; on triplets of symbols we have however the following situation: if a triplet ends in $\otimes$ then $h = 3$ except for $h(\otimes \otimes \otimes) = 0$; if a triplet ends in $\cdot$ or $-$ then h is 2 or 1 for some of them. This situation allows us to single out $\otimes$ as an end letter and we proceed as follows: we rewrite Λ^1 by changing each $\cdot$ that is followed by $\otimes$ to α and each $-$ that is followed by $\otimes$ to λ and apply Procrustes. Taking $h_0 = 5$ we find that Σ^1 coincides with Λ^1 except that it contains the additional string $-\cdot\lambda\otimes = -\cdot-\otimes$ which corresponds to the letter K. Another interesting situation occurs if we originally were to consider that both $\cdot$ and $\otimes$ can serve as end letters, then changing each $-$ that is followed by $\cdot$ or $\otimes$ to λ we find using $h_0 = 4$ that Σ^1 contains the following strings: $\{\cdot, B, C, D, G, K, M, N, O, Q, T, Z\}$ and that starting from Σ^1 using $h_0 = 12$ we obtain as Σ^2 precisely Λ^1. The letter K disappears because the texts make no use of it.

The above procedure was one way to treat the case when h is constant. It consists of singling out a proper subset I of Σ^0 that is <u>not</u> to become an end letter and of rewriting the text by changing each letter λ in I to some new one if it precedes a letter not in I. We apply Procrustes now to this new situation. For another procedure see [5][2].

Concerning the optimality of each transition from Σ^i to Σ^{i+1} we can say the following. If we take h_0 to be the largest possible that secures a satisfactory convergence then Σ^{i+1*} is the smallest possible and this can be deemed to be optimal. On the other hand, for a very detailed resolution of the hierarchical structure it may be a propos to take the smallest h_0 that gives some syllabification. This provides for very many levels which should prove useful for detailed studies.

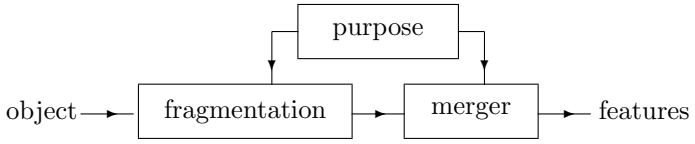
Unfortunately at the time of writing we cannot give an example from an actual natural text, but as we are now writing a computer program for the algorithm we should have abundant examples shortly.

We now make a few general comments that constitute departure points for further research.

(1) The duality existing between feature and purpose or motivation is a sorely neglected question. On the psychological level we know that perception

[2]Unfortunately this reference is lost.

and motivation influence each other in a tight dynamical relationship. This relationship must be taken into account when constructing motivated machines that must perform certain tasks not knowing what difficulties they will encounter along the way. Such a machine if encountering a difficulty must not only be able to analyze what is impeding it but must also be able to subject its own thinking procedure to analysis thus subjugating certain purposes to higher ones. This can be made more explicit in terms of the two basic processes. The merger process has as its output a set of features; the input can be considered to consist of two types, the first is the object under consideration, the second is the principle of merger. This latter one can be identified with the purpose toward which we perform the mergers. A similar statement can be made concerning the fragmentation process. Schematically a basic unit of our motivated machine is the following.



Now the whole operation of the above unit can be subjected to another such, leading therefore to a hierarchy of purposes. In terms of the Procrustes algorithm the purpose input corresponds to the choice of h_i^0. A fragmentation of the algorithm would correspond to choosing a set of cuts say all the integers between and including the minimum and the maximum of h on Σ^i. The outputs of all of these algorithms can then be subjected to by another merger machine which would now be extracting the features of the algorithm-language complex rather than the features of the language,and on the basis of its output choose the optimal h_0 that suits its purpose. A whole array of such motivated units can be envisaged interconnected in various manners giving rise to very complex machines. Such machines can readjust their feature extraction processes to correspond to the task at hand.

(2) The concept of a progressively freer hierarchy, though clearly applicable to natural languages covers a much wider class of situations. One could of course mention gesticulated languages but even more generally the whole behavioral repertoire of organisms in an unconstrained environment has this feature. A cat free to explore a room undergoes a certain sequence of bodily positions. If taken at extremely short time intervals they show a very rigid structure governed by basic anatomical dynamics. On a higher.level the cat has a certain repertoire of behavior which it can combine in any sequence with short transitional behavior. This may not be the highest level for a particular cat can have favorite sequences expressing its personality. A new class of behavioral studies can eventually be opened up from this point of view.

Another possible example of a progressively freer hierarchy is the chemical composition of organisms. The smallest chemical units are very restricted and are common to all organisms but the larger units seem to combine in a freer

fashion leading to vast variety of structures. The study of protein sequences from this point of view can lead to new insights and is being initiated.

(3) The Procrustes algorithm as described above is local in that the decision of whether to keep a string or not depends only on the immediately following letters and ignores all other strings. The syllabification of Σ^i into Σ^{i+1} so obtained should however have a global characterization. The syllabification given by the algorithm is the optimal one in some sense and there should exist some well defined information theoretic number, which is a function of the text and the syllabification of that text, and which measures this optimality in that it takes its maximum value at the Procrustes syllabification. Such a global characterization would allow immediate generalization to situations that are not linear such as a text but to a more general combination of features. In this manner we hope to obtain merger algorithms in the most general situations.

The author is indebted to Professors E. R. Caianiello, R. M. Capocelli and A. Restivo for helpful discussion. Note: After this paper was written the author came across a work [3] that is very close in spirit to the Procrustes algorithm but works on pictures rather than texts.

[3] Unfortunately this reference is lost.