UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament de Llenguatge i Sistemes Informàtics
Ph.D. Programme: Artificial Intelligence

SYMBOLIC AND CONNECTIONIST LEARNING TECHNIQUES FOR GRAMMATICAL INFERENCE

Autor: René Alquézar Mancho
Director: Alberto Sanfeliu Cortés

March 1997

Chapter 1

Introduction

In this first chapter, the location of the thesis in the field of artificial intelligence is established, the subject of grammatical inference is introduced, the objectives of the work are presented, and the structure of the thesis is described.

Firstly, the concepts of learning and inductive inference are defined, and some background about the theory of inductive inference is provided. In particular, Gold's general paradigm of inductive inference is recalled, the elements involved in an inductive inference problem are discussed, and some general methods (enumeration, best guess) are described.

Grammatical inference (GI) is then defined with respect to the inductive inference paradigm, and the different types of input data presentations that are possible in a GI problem are commented. Some theoretical limitations of GI are remembered, and the availability of GI methods for the classes of formal languages in the Chomsky hierarchy is analysed. The relationship between grammatical inference and syntactic pattern recognition is discussed, and some applications of GI are mentioned.

A very brief introduction to artificial neural networks is also given, putting emphasis on their use for grammatical inference. Some connectionist GI approaches are recalled and interpreted with regards to the inductive inference paradigm.

1.1 Learning and inductive inference

During the last thirty years, it has become apparent that *learning* is a foremost topic within the field of *artificial intelligence* and a great deal of effort has been devoted to it by researchers [MiCM:83, MiCM:86, OsSW:86, KoMi:90, Kodr:92]. To learn can be defined as "to gain knowledge, understanding, or skill by study, instruction, or experience". The ability of learning is clearly demanded to any machine that could be considered "intelligent", in front of the traditional "silly" machines that are programmed in detail to solve each task. Even in the so-called knowledge-based systems, where the emphasis is put not in how to perform tasks but in the knowledge required to do so, the problem of automating the knowledge acquisition is of outmost importance.

The topic of learning has been divided into several areas as progress in artificial intelligence has developed. The survey article by Dietterich *et al.* in the former eighties [DiLC:82] included four areas: *rote learning*, *learning by being told*, *learning from examples*, and *learning by analogy*. Since then some areas have been renamed or redefined: *explanation-based learning* (EBL) [MiKK:86] is related to learning by being told, *inductive inference* [AnSm:83,Kodr:92] includes learning from examples, and *case-based reasoning* (CBR) [KoSS:85] covers learning by analogy. Furthermore, new areas have been incorporated and explored, the most important being learning by *artificial neural networks* (ANNs) [RuMc:86,HeKP:91], learning by *genetic algorithms* (GAs) [Goldb:89,DeJo:90], and *reinforcement learning* for behavior-based robots [MaCo:92]. In the more recent review by Kodratoff [Kodr:92], machine learning methods were grouped into three general classes, according to the type of inference used, i.e. deduction, induction or analogy, and most of the learning methods reported in the literature were located in the inductive inference group (e.g. version space [Mitc:82], star algorithm [Mich:83], conceptual clustering [StMi:83], decision tree induction [Quin:86], and also the genetic and connectionist learning approaches).

Inductive inference is a way of learning that can be defined as "the process of inferring general rules from examples, or theories from facts, carried out by an agent or system". Traditionally, there has been two branches in inductive inference research [AnSm:83]: the study of the general theoretical properties of inference methods on one hand, and the design of specific practical inference methods on the other hand. However, in the latest years, a growing tendency has been to bring nearer both lines and to design algorithms that are well-supported by the underlying theory (e.g. [GaVi:90,OnGa:92]).

A great part of the work on the theory and methods for inductive inference has been done within the general paradigm of inductive inference established by Gold

in a fundamental paper [Gold:67]. In this paradigm, an agent tries to identify a rule or object (from a determined class of rules or *object space*), subject to a certain *success criterion*, by collecting *data* (either obtained by passive observation or active experimentation) and guessing a description of the rule, which is selected from a determined *hypothesis space* using the data. More precisely, the following five items must be specified to define an inductive inference problem [Gold:67,AnSm:83]:

(1) the *object space* or class of rules being considered, which is usually a class of functions or languages;
(2) the *hypothesis space*, which is a set of descriptions such that each rule in the class has at least one description in the hypothesis space;
(3) for each rule, its set of *examples* (typically, points of the function graph or strings over an alphabet), and the sequences of examples that constitute *admissible presentations* of the rule;
(4) the *class of inference methods* under consideration; and
(5) the *criterion* for a successful inference.

Another item has been appended by Pitt to the above list [Pitt:89]:

(6) other means (if any) by which the inference method can obtain additional information about the rule to be inferred.

In particular, a learning algorithm might have access to an *oracle* that answers specific kinds of *queries* about the unknown target object [Angl:88].

If only examples are used, the class of inference methods considered is, in general, the class of all algorithms that take as input a finite initial segment of an admissible presentation, always halt, and produce as output a hypothesis from the hypothesis space. However, several restrictions can be posed on inference methods that will be commented later; e.g. a *consistent* method must always yield hypotheses that are compatible with the examples read in so far. A variation of this scenario is the case of *given data* [Gold:78], when the set of available learning examples is fixed, and the learning algorithm proposes just one hypothesis.

Another variation is the case of *learning using queries*, where the algorithm starts with a (possibly null) set of examples and then can ask a finite number of queries of certain types to an oracle before halting and proposing an hypothesis. Here, the object space is assumed to be a countable collection of subsets of a universal set. The simplest type of queries are *membership* queries, where the inference method asks the oracle whether a certain element belongs or not to the target subset, but other types of queries have been described and studied: *equivalence*, *subset*, *superset*, *disjointness* and *exhaustiveness* queries [Angl:88].

1.2 Success criteria for inductive inference

The theoretical success criteria proposed in the literature for inductive inference problems are briefly reviewed in this section: *identification in the limit*, both in the cases of sequential presentation [Gold:67] and given data [Gold:78], *behaviorally correct identification* [Feld:72], EX^k-*identification* [CaSm:83], *approximate identification* [Whar:74], PAC-*identification* [Vali:84]. On the other hand, the correct generalization performance of a selected hypothesis on a test data set is usually taken as a practical success criterion.

The most classical theoretical success criterion is (exact) *identification in the limit* (or EX-*identification*) [Gold:67], which views inductive inference as an infinite process and is based on the limiting behavior of the inference method. Its traditional formulation is as follows:

Let M be an inductive inference algorithm that is attempting to describe correctly some unknown target rule R. To that end, M is supplied with a growing sequence of examples of R (in an admissible presentation), say $D_1, D_2, ..., D_i, ...$, and at each step i, M proposes a hypothesis H_i (using D_i) that represents the guessed description of R at this step. In this way, by running M repeatedly on larger and larger collections of examples, an infinite sequence of hypotheses $H_1, H_2, ..., H_i, ...$ is generated. If there exists some finite step t such that H_t is a correct description of R and from this step all the guessed descriptions are the same (i.e. $H_i = H_t$ for $i > t$) then M is said to identify R correctly in the limit on this sequence of examples. The algorithm M is said to identify in the limit a certain object space, if M is able to identify in the limit every rule R of the object space on every admissible presentation of its examples. Less formally, an algorithm is said to have the *identification in the limit* property if it always converges to a correct hypothesis describing the target object, when provided with a growing sequence of data.

It is important to note that M cannot determine whether it has converged to a correct hypothesis, even if it has the identification in the limit property, since new data may or may not conflict with the current guess. In order to become aware of the convergence to a correct hypothesis, a learning algorithm would need to ask periodically an equivalence query to an oracle, which would eventually confirm the identification.

In the preceding scheme, it is supposed that as much data as needed is available or can be requested. Another formulation of identification in the limit has been given by Gold for the case of *given data* [Gold:78], when the set of available examples D is fixed. In such a case, the algorithm M proposes a hypothesis H for the unknown R using D. The algorithm M is said to identify in the limit an object space, if, for every

rule R of the object space, it is possible to define a finite set of examples D_R^r such that for every set D containing D_R^r the corresponding hypothesis returned by M is a correct description of R. The set D_R^r is called a *representative sample* for the rule R, but it not only depends on R but also on the inductive inference algorithm M. Again, an algorithm with the identification in the limit property cannot know whether the returned hypothesis is correct or not, unless an oracle tells it.

Identification in the limit is also called EX-*identification*, and EX denotes the class of all the object spaces U such that some algorithm M identifies U in the limit. A fundamental result, due to Gold [Gold:67], is that the set of all total recursive functions cannot be identified in the limit by any inductive inference method; likewise, the set of all recursively enumerable languages, those languages accepted by Turing machines, cannot be EX-identified either. The success criterion defined by identification in the limit can be weakened in several ways, thus allowing more classes of rules to be effectively identifiable.

The *behaviorally correct identification* (or BC-*identification*) [Feld:72] requires that after some finite initial segment, all the guesses be correct descriptions of the underlying rule (i.e. the hypotheses could not converge, but the described rule must do). Another weakening direction is to permit the final guess to be "nearly correct" as a representation of the target rule. Two such success criteria are: EX^k-*identification* [CaSm:83], or *identification in the limit with at most k anomalies*, and *approximate identification* [Whar:74].

An algorithm M *identifies in the limit with at most k anomalies* (or EX^k-*identifies*) a rule R if M converges to a hypothesis that describes a rule differring from R in at most k elements (points for functions or strings for languages). EX^k denotes the class of all object spaces U for which some algorithm M EX^k-identifies all the rules in U. EX^*-*identification* permits any finite number of anomalies, and the class EX^* can be defined similarly. Then, $\mathrm{EX} = \mathrm{EX}^0$, and EX^0, EX^1, EX^2,... is a proper hierarchy of classes, the union of which is contained in EX^* [CaSm:83]. Anomalies may also be permitted with respect to the behaviorally correct identification criterion, thus defining analogously the BC^k, for $k = 0, 1, 2, ...$, and BC^* classes, where $\mathrm{BC} = \mathrm{BC}^0$, and again a proper hierarchy is obtained [CaSm:83].

If the anomalies are weighted in a probabilistic sense, the criterion of *approximate identification* proposed by Wharton [Whar:74] can be followed for language inference (or, in general, for the inference of subsets of a universal set U). This success criterion is based on the following notion of approximation of one language by another one. Firstly, it is assumed that each string s over the alphabet is assigned a certain probability $p(s)$, so that the sum of all the string probabilities is 1. Then the distance $d(L_1, L_2)$ between two languages L_1 and L_2 is defined as the sum of the probabilities of the strings in the

symmetric difference of L_1 and L_2. Then, given a positive parameter ϵ, an algorithm M *ϵ-identifies* a language L (or approximately identifies L with accuracy ϵ) if and only if it converges to a hypothesis h such that $d(L(h), L) < \epsilon$. It can be proved that, for any positive number ϵ, the class of all languages over a given alphabet is ϵ-identifiable by an algorithm that produces only regular grammars.

A related criterion of probabilistic identification was introduced by Valiant, the *probably approximately correct identification* (or PAC-*identification*) [Vali:84]. In this case, a sampling process draws elements (examples) of a universal set U according to a probability distribution D defined on U and supplies them to a learning algorithm. An algorithm PAC-*identifies* a target subset L with accuracy ϵ and confidence factor δ if it converges to a hypothesis h and we have a probability lower than δ that the distance $d(L(h), L)$ is greater than ϵ, where the distance between two subsets is defined as before. That is, with high probability there is not too much difference between the conjectured set and the target set.

In the case of inferring stochastic languages, the usual success criterion is *identification in the limit with probability* 1, which filters out the effects of statistically aberrant presentations. Interestingly, some classes of languages (e.g. context-free and regular languages) are not identifiable in the limit from positive presentations, but their stochastic counterparts are identifiable in the limit with probability 1 from stochastic presentations, which includes no explicit negative example [Horn:69].

However, the above success criteria are mainly of theoretical interest, since in most practical situations we cannot expect to receive an arbitrarily long sequence of examples or to have available an oracle answering an arbitrarily large number of queries. To the contrary, the assumption of a set of *given data* is much more realistic. Often in this case the hypothesis space can be organized in a helpful search space structure (e.g. a lattice), allowing the elimination of more hypotheses than just one when an incompatibility with the examples is detected [Mitc:82, MiGe:94]. However, even if pruning techniques are applied, a systematic search of the hypothesis space, to obtain the most general or the simplest solution, normally takes exponential time. Any practical algorithm must run in polynomial time, and therefore, most practical inductive inference methods have used some *heuristic* criterion to select a hypothesis within the set of hypotheses consistent with the data. Nevertheless, some practical methods have been proposed that are both efficient (polynomial-time) and yield *characterizable* results; moreover, they identify in the limit a certain class of sets or languages [AnSm:83,GaVi:90]. Either if a heuristic or a characterizable inductive inference method is applied to a learning set of given data, the usual way of assessing the goodness of the returned hypothesis is by confronting it with a test data set different from the learning set.

1.3 Restrictions on inductive inference methods and some general methods.

Besides halting and performing a mapping from admissible presentations to hypotheses, which are obvious requirements, other restrictions may be placed on the algorithms to be used as inductive inference methods. In general, the effect of a restriction is to reduce the classes of rules identifiable, since some possible algorithms may not satisfy the restriction. A list of some common restrictions together with their respective meanings is given below.

- *finite identification*: as soon as a hypothesis is repeated at two successive steps, the algorithm halts, that is the algorithm must finitely converge to a correct hypothesis;
- *reliability*: if the algorithm converges on an input sequence, it must converge to a correct hypothesis;
- *consistency*: the hypotheses yielded by the algorithm must be consistent (or compatible) with the examples read in up to the time that the guess is made;
- *hypothesis conservation*: an algorithm is conservative if it outputs a hypothesis different from its previous one only when the previous guess is incompatible with the examples read in so far;
- *class-preservation*: an algorithm is class-preserving on an object space U if it only outputs hypotheses that describe rules in U;
- *optimal hypothesis size*: the algorithm must produce as ultimate guess the "smallest" possible description among the correct hypotheses;
- *optimal data efficiency*: an algorithm is optimally data efficient if there is no other method that requires less data to converge to a correct hypothesis (for all possible presentations of rules);

It turns out that finite identification is much harder than identification in the limit. There also exist object spaces that are identifiable in the limit, but not by any consistent method [AnSm:83].

Identification by enumeration is an inductive inference general method that is reliable, consistent, conservative, class preserving and optimally data efficient. It consists of systematically searching the space of possible rules until one is found that agrees with all the data so far. Given a certain object space, this method requires that there exist an enumeration of descriptions, say, $d_1, d_2, ..., d_i, ...$, such that each rule in the object space has one or more descriptions in this enumeration. Given any collection of examples of a rule, the method of identification by enumeration scans this list to

find the first description, say d_i, that is compatible with the given examples and then outputs d_i. We may think of identification by enumeration as a general search method that can be adapted to different domains by giving it different generators for the spaces to be searched.

The enumeration method may or may not achieve correct identification in the limit, and may or may not be computable. It is guaranteed to identify in the limit all the rules in an object space if the following two conditions (on the example presentations and the compatibility relation) holds:

i) any correct hypothesis is always compatible with the examples given;
ii) any incorrect hypothesis is incompatible with some sufficiently large collection of examples and with all larger collections.

The enumeration method is computable if the enumeration $d_1, d_2, ..., d_i, ...$ is computable and it is always possible to compute whether a given description and a given collection of examples are compatible or not.

Identification by enumeration is a very general and powerful method but it is usually impractical because the size of the space that must be searched is typically infinite or exponential in the length of the description of a correct hypothesis. Whenever an inductive inference problem satisfies the two aforementioned conditions, defining a computable enumeration method for it serves to prove that a certain object space is identifiable in the limit. Thus, Gold proved that all the classes of languages that are generated by a recursively enumerable class of grammars (e.g. context-sensitive grammars) are identifiable in the limit from a complete presentation [Gold:67], since it is possible to check orderly whether each grammar in the enumeration generates or not the presented strings (i.e. to check its compatibility with the data).

Another general inductive inference method is *identification by best guess.* It is based on defining some measure of goodness of hypotheses with respect to samples, and using some method of finding the hypotheses that are optimal with respect to this measure. The fundamental aim is to guarantee the quality of the hypotheses produced at every step, and not only in the limit. The usual measures of goodness are related to the *simplicity* of the hypothesis and its *goodness of fit* to the finite sample seen so far.

Any such measure of goodness can be represented as a partial ordering, called a *goodness ordering*, of the hypotheses compatible with a given sample S. Thus, a relation $g < h$ *with respect to* S can be defined partially that indicates that a hypothesis g is a better explanation of the sample S than the hypothesis h, according to the chosen measure. For some pairs of hypotheses (g, h), it may occur that neither $g < h$ nor $h < g$, so g and h are incomparable in the ordering and none of them is considered a

better explanation of the sample than the other. The minimal elements in a goodness ordering constitute the set of *best guesses* for a given sample S.

A goodness ordering is *computable* if there is an algorithm A that, for every finite sample, it returns an element of the set of best guesses for the sample. And a goodness ordering is *sample independent* if and only if whenever $g < h$ with respect to some sample S, then $g < h$ with respect to every sample with which they are both compatible. Obviously, the orderings based only on some property of the hypotheses, like size, are sample independent.

A general inductive inference method P, which needs a computable sample-independent goodness ordering $<$ and an associated algorithm A for finding a best guess for a given sample, can be defined as follows. Firstly, a best guess for the initial sample is obtained as current hypothesis using A. Then, the method continues to read data and check the compatibility of the current hypothesis. While the current hypothesis remains compatible, it is not changed, but if it becomes incompatible with the data, a best guess for the current sample is computed using A and taken as the new current hypothesis.

The above method P is conservative and guarantees that, at every finite stage, the current hypothesis is a best guess for the sample seen so far. Furthermore, depending on the properties of the specific domain and goodness measure, it may correctly identify in the limit some object spaces, as in the cases commented next.

Given a certain measure of simplicity on the set of hypotheses, we can consider that $g < h$ if g is simpler than h. This is a sample-independent goodness ordering, such that the set of best guesses for a sample S is the set of simplest hypotheses that are compatible with S.

For instance, given a sample $S = (S^+, S^-)$ containing positive and negative examples (strings) of a regular language, one might be interested in finding a finite-state automaton (FSA) with the fewest possible states that accepted all the strings in S^+ and none of the strings in S^-, where in this case, the number of states is the measure of simplicity of the hypotheses (FSAs). An algorithm A that computes the minimal FSA compatible with a sample can be devised, which consists of enumerating FSAs in order of increasing numbers of states and taking the first compatible one. If this algorithm, or any other algorithm A that found a minimal compatible FSA, were used in the preceding inductive inference method P, then P would correctly identify in the limit all the regular languages. Unfortunately, the computational problem of finding a minimal FSA compatible with given data is NP-hard [Gold:78], and therefore, any known algorithm A for this problem requires exponential time on some inputs.

If regular expressions were considered as hypotheses, and the size of a regular expression (RE) is defined to be the number of occurrences of alphabet symbols it contains, then the inductive inference method P using an algorithm A that computed an RE of minimal size compatible with a sample $S = (S^+, S^-)$ would again identify in the limit any regular set. However, the problem of finding a regular expression of smallest size compatible with a given sample $S = (S^+, S^-)$ is also NP-hard [Angl:78].

Given a hypothesis space H such that each hypothesis $h \in H$ describes a language $L(h)$ in a certain class of languages, we can consider that $g \leq h$ if and only if $L(g) \subseteq L(h)$. This is a sample-independent goodness ordering, and the set of best guesses for a sample S^+ containing only positive examples is the set of hypotheses H_{bg} that best fit to the sample S^+, i.e. for all $h \in H_{bg}$, $S^+ \subseteq L(h)$ and there is no $g \in H$ such that $S^+ \subseteq L(g) \subset L(h)$. The inductive inference method P can be applied if an algorithm A returning a best fit hypothesis for any given S^+ is available (since the set inclusion ordering is sample-independent). The use of the scheme P with this type of goodness of fit measure leads to the correct identification in the limit of certain classes of languages from positive examples [Angl:80a], and furthermore, efficient algorithms A to be used in the method P are known for some specific domains [CrGM:78, Angl:80b, Angl:82, GaVi:90].

Some other goodness measures have been proposed that combine measures of hypothesis simplicity and goodness of fit to the data. A particular way of achieving this kind of combination is by using a Bayesian approach in which a hypothesis that maximizes the conditional probability $Pr(h|S)$ of a hypothesis given the observed sample is looked for [AnSm:83]. By Bayes theorem,

$$Pr(h|S) = \frac{Pr(h)Pr(S|h)}{Pr(S)},$$

and therefore, to maximize $Pr(h|S)$, it suffices to maximize $Pr(h)Pr(S|h)$, where $Pr(h)$ can be thought of as a measure of the simplicity of the hypothesis h (assuming higher probabilities for simpler hypotheses) and $Pr(S|h)$ as a measure of the goodness of fit of h to the sample S (such that a higher probability corresponds to a better fit). However, the inductive inference method P is not generally applicable with such a probabilistic ordering, since this type of ordering is usually not sample-independent. On the other hand, enumerative (inefficient) methods for finding a hypothesis h that maximizes $Pr(h|S)$ have been reported that identify in the limit with probability 1 the classes of stochastic context-free grammars [Horn:69] and stochastic regular grammars [VaWa:78], respectively, from stochastically generated positive samples.

In addition to the matter of guaranteeing a good quality of the hypotheses proposed by an inductive inference method, which is related to the goodness measures and the best guess approaches, another relevant issue concerns the efficiency of the method,

which is usually related to the requirement of using polynomial time and space. Thus, a desirable property of an inductive inference method is *polynomial inference*, that combines the properties of identification in the limit and polynomial complexity. However, different definitions of polynomial inference have been given in the literature [Gold:78, Angl:87, Pitt:89, DelH:96].

Within the paradigm of sequential presentation, Pitt defined that an identification algorithm is polynomial if it has polynomial update time (to propose a new hypothesis, with respect to the size of the sample seen so far) and makes at most a polynomial number of implicit errors (with respect to the size of the target hypothesis) [Pitt:89]. An implicit error is made when the current hypothesis does not agree with a new example. Pitt's definition has been shown to be quite restrictive, and a looser definition of polynomial inference have been proposed for the case of learning from given data [Gold:78, DelH:96].

Thus, De la Higuera [DelH:96] defines that an algorithm A *polynomially identifies* (the class of objects described by) the hypotheses in H *from given data* if and only if there exist two polynomials $p()$ and $q()$ such that

i) given any sample S of size m, A returns a hypothesis in H that is compatible with S in $p(m)$ time; and

ii) for each hypothesis $h \in H$ of size n, there exists a characteristic sample CS of size less than $q(n)$ for which, whenever the sample S contains it, A returns a hypothesis equivalent with h.

The size of a sample is taken as the sum of the sizes of the included examples. A certain hypothesis space H is said to be *polynomially identifiable from given data* if there exists an algorithm A that polynomially identifies H from given data [DelH:96].

The inductive inference methods based on systematic search and pruning of hypothesis spaces tend to be very time consuming, being typically of exponential complexity. Note also that the property of being polynomial does not necessarily mean the same as practical. In order to be actually practical, an algorithm should run in time $O(n^k)$ for a small value of k, say $k \leq 3$. Hence, efficient *constructive methods* have been proposed for practical learning systems, such that just one or a few hypotheses are directly built from the given data based on a set of heuristic criteria and/or subclass assumptions. As has been pointed out in the previous section, constructive methods are usually classified in two groups: *heuristic* and *characterizable inference methods* [AnSm:83]. They are distinguished by the fact that the methods in the latter group allow a formal description of their results (i.e. of the constructed hypotheses). Normally, the characterizable methods are associated with subclass assumptions, such that the inference result can be shown to represent the smallest language in the restricted subclass of languages that contains the given sample.

Knuutila has recently proposed a general framework based on monotonic representation mappings to build characterizable inference methods for different domains [Knuu:96]. Likewise, some general schemes have been described that try to cover a variety of heuristic inference methods for specific object spaces, e.g. for the regular languages [KuSh:88, Knuu:96].

1.4 Grammatical inference

The object spaces considered in the present work are classes of languages. A *class of languages* is a collection of subsets of a universal set, that is formed by all the strings over a given alphabet of symbols. The hypothesis space for a class of languages is typically a class of formal grammars (e.g. the regular grammars) or a class of acceptors (e.g. the finite-state automata), although other types of representations are possible (e.g. the regular expressions) [Salo:73, HoUl:79].

The term *grammatical inference* (GI) [FuBo:75, Fu:82, Micl:90, Vidal:94, MiDH:96] has been traditionally used to denote the inductive inference of classes of languages described by grammars. Nevertheless, this does not mean that the hypothesis space for a GI method is necessarily constituted by grammars, but for any available type of descriptors for the involved class of languages (e.g. certain types of automata, transition networks, regular expressions, ...). Furthermore, in a relaxed sense, the meaning of the term *grammatical inference* can be extended to include the inductive inference of whatever well-defined class of languages, even if a precise class of grammars is not known for it.

In the case of inferring an unknown language L, there is an important distinction between giving only positive information (i.e. members of L) and giving both positive and negative information (i.e. both members and nonmembers of L). Gold introduced two types of admissible presentations of a language L [Gold:67]: a *positive presentation* of L is an infinite sequence giving all and only the strings belonging to L; a *complete presentation* of L is an infinite sequence of ordered pairs $\langle s, d \rangle$ such that s is a string, $d = 1$ or 0 depending on whether s is a member of L or not, and every possible string over the given alphabet appears as the first component of some pair in the sequence. Gold also proved [Gold:67] that

(1) all the recursively enumerable classes of languages (context-sensitive and below) can be identified in the limit using a complete presentation; and, on the other hand,

(2) no *superfinite* class of languages, i.e. one that contains all finite languages and at least a language that is infinite (e.g. the regular languages), can be identified in the limit using a positive presentation.

However, there are certain (non superfinite) subclasses of regular languages that can be identified in the limit from only positive examples using polynomial algorithms, e.g. the classes of *k-reversible* languages [Angl:82], *k-testable* languages [GaVi:90], and *terminal distinguishable regular languages* [RaNa:87]. The conditions that are required to identify in the limit a class of languages from just positive data were stated by Angluin [Angl:80a]:

Let $\mathcal{C} = \{L_1, L_2, ...\}$ be an indexed class of languages. Class $\mathcal{C}$ is identifiable in the limit from a positive presentation if and only if there exists an algorithm that on any input $i \geq 1$ enumerates a finite set of strings $T_i \subseteq L_i$ such that $\forall j \geq 1 : T_i \subseteq L_j \Rightarrow L_j \not\subset L_i$. The sets T_i tell each language L_i from any other language L_j, $i \neq j$. Actually, L_i is the smallest language in the family $\mathcal{C}$ containing the set T_i. The underlying idea is that by obtaining more and more examples a characteristic sample will eventually be formed containing all the strings needed to identify the target language. Other conditions on both the class of languages and the hypothesis space that guarantee the identification from positive data have been stated recently by Knuutila [Knuu:96].

Concerning the cases of identification from complete presentations (both positive and negative examples), although one of Gold's theorems establishes that some important classes of languages (e.g. context-sensitive, context-free, linear and regular languages) can be identified in the limit, it is important to know whether these classes of languages can be polynomially identified or not from complete presentations. Unfortunately, the most part of results proved in the theory are quite negative in this matter. For example, if Pitt's definition for polynomial inference is adopted, then it results that neither the class of regular languages nor any superclass of it can be polynomially identified in the limit from a complete presentation [Pitt:89]. If the definition of polynomial inference from given data is taken, then neither linear nor context-free (nor higher level) languages can be polynomially identified from given data, but, in this case, the regular languages can be, whenever deterministic finite-state automata (DFA) are chosen for the representation space[1] [OnGa:92a-b,DelH:96]. The class of even-linear languages has also been proven polynomially identifiable from given data [Taka:88,SeGa:94].

Another way of presenting a language L is *presentation by informant* [AnSm:83], where an informant is an oracle that the inference method may ask membership queries about L. In a very abstract sense, presentation by informant is equivalent to complete presentation, but in practice algorithms using queries are rather different from algorithms using examples and no queries. Angluin proved that, using only

[1] De la Higuera have demonstrated that, under the assumption P $\neq$ NP, the class of non-deterministic finite-state automata (NFA) is not polynomially identifiable from given data [DelH:96].

membership queries, regular languages cannot be polynomially identified, with time complexity depending on the size of the target minimal DFA; but on the other hand, if a structurally complete sample is provided in advance, then regular languages can be identified rather efficiently using a polynomial number of membership queries [Angl:81]. If both membership and equivalence queries are allowed (a so-called *minimally adequate teacher* (MAT) model), then regular languages can be identified using a polynomial number of queries and in time polynomial with respect to the length of the longest counterexample returned in an equivalence query [Angl:87]. Likewise, the class of *simple deterministic languages*, which is a subclass of the context-free languages, has been proven to be polynomially learnable with a MAT model as well [Ishi:89]. Other types of presentations of languages based on further classes of queries answered by an oracle have also been studied [Angl:88].

Other object spaces that have been considered in some GI problems are the classes of *stochastic languages*. A stochastic language is a language together with a probability distribution on its elements. The corresponding hypothesis space is usually a class of stochastic grammars, although it may also be a class of stochastic automata. A stochastic grammar has probabilities associated with alternative productions, which are used to define the probability that the grammar derives a given string. An example of a stochastic language is an element of the language (i.e. a string), and an admissible presentation is any infinite sequence of examples such that the strings in the presentation are assumed to be generated by random selection from the language according to the probabilities associated with its elements. This kind of presentation is called a *stochastic presentation*, and any finite subset of it is called a *stochastic sample*.

Inference of stochastic languages from stochastic presentations has been studied in several works either using direct or indirect methods [Horn:69, CoRA:76, Fu:82, ThGr:86]. In the latter case, a two-step approach can be followed: to infer firstly a characteristic grammar or automaton, without probabilities, using a (not necessarily stochastic) GI algorithm, and then to infer the corresponding probabilities of the rules or transitions using a *probability estimation* technique [LaYo:91, Casa:94]. For inferring stochastic regular languages, a probability estimation technique of *Hidden Markov Models* (HMMs) [Rabi:89] can be applied directly to the learning examples whenever the number of states in the target automaton is known or estimated in advance.

As a concluding remark, it can be seen that if all the practical GI methods found in the literature (including polynomial identification, characterizable and heuristic methods) are classified according to the expressive power of the inferred languages, we realize that the most part of them infer regular languages or some subclass of regular languages [Micl:90, Vidal:94, Greg:94]. Much less methods are known for learning context-free languages, and the ones that cover the largest subclasses require structural descriptions of the positive examples in the form of unlabeled parse trees [CrGM:78,

Saka:92]. Finally, there have been very few reported methods to infer descriptions of context-sensitive languages; as far as I know, an old semi-automated heuristic procedure for inferring augmented transition networks (ATNs) by Chou and Fu [ChFu:76], and a rather recent work by Takada on the inference of some families of controlled even-linear grammars learnable by regular language learning algorithms [Taka:94].

1.5 Neural networks and their use for grammatical inference

The term *neural network* originally referred to a network of interconnected neurons, but currently, this term, or more properly the term *artificial neural network* (ANN), has come to mean any computing architecture that is composed of many interconnected simple computational elements ("neural" processors) operating in parallel. ANN models have received other alternative names such as *connectionist models* and *parallel distributed processing models.* Most ANN architectures are reminiscent of biological neural nets and are based on the present understanding of biological nervous systems.

The motivation for the study and development of ANNs in the field of artificial intelligence comes mainly from the fact that humans are much better at pattern recognition tasks like speech and image recognition than current computer systems. The brain has taken millions of years to evolve into its present architecture and has become the most efficient machine in some processes like vision. Hence, there must be computational principles that the brain uses to accomplish such high speed pattern recognition, which may be applied in the design of practical systems.

ANN models are specified by the *net topology*, the characteristics of the computational unit or *neuron*, and the training or learning rules (*neural learning algorithms*). The neurons are typically connected via *weights* that are adaptable. Normally, the learning rules specify an initial set of weights (maybe random) and indicate how weights should be adapted during use to improve performance [RuMc:86]. Some learning algorithms also construct the structure of the network (*constructive methods* [FaLe:90]) or prune the connections among the neurons during training (*pruning methods* [Reed:93]).

Although neural networks can serve to deepen the understanding of brain functions, engineers and AI practitioners are interested in neural networks for problem solving. As an engineering technique, ANNs have their own set of advantages and drawbacks. Certainly, the high computation rates provided by massive parallelism are a first benefit. Also, because of their adaptive nature, ANNs can adapt to changes in the data and learn. In addition, ANNs typically provide a greater degree of robustness or fault

tolerance than traditional systems, since, due to the many processing nodes and local connections, damage or removal of a few neurons or connections does not necessarily impair the overall performance significantly. Furthermore, because of their nonlinearity, ANNs can perform functional approximation and signal filtering operations which are beyond optimal linear techniques. Finally, ANNs can be used in pattern classification since they can adjust nonlinear and nonconvex regions in the feature space.

On the other hand, multiple minima of the cost function can appear during neural learning and there is no guarantee that a global minimum associated with an optimal performance will be finally reached. Although some stochastic techniques like simulated annealing may help in this problem, their application typically requires a huge computation time. Moreover, the simulations of ANNs in digital computers are still too slow for practical use in large scale problems. Concerning understandability, the knowledge learned and represented by ANNs is often obscure and difficult to translate into symbolic terms for human comprehension.

Work on ANN models has a long history. The development of the earlier models was originated from the aim of building brain-like computers out of neuron-like parts. Already in the 1940s, McCulloch and Pitts modeled a neuron as a simple threshold device to perform logic function [McPi:43]. By the late 1950s and early 1960s, the *perceptron* [Rose:62] and *adaline* [WiHo:60] models were proposed for classification and adaptive signal processing, together with their corresponding learning rules, namely, the perceptron convergence procedure [Rose:62] and the LMS (least mean square) algorithm [WiHo:60], respectively. Later, in the late 1960s, Minsky and Papert showed that the perceptron can only classify linearly separable regions and cannot be used for complex logic functions [MiPa:69], and this work caused a notable decrease in ANN research. During the 1970s, just a few investigators continued working on the field: Kohonen was developing the *self-organizing maps* (SOM) based on the physiological principle that neurons in the sensory pathways in the brain often organize themselves to reflect some physical characteristic of the external stimulus being sensed [Koho:84]; and Grossberg was developing the *adaptive resonance theory* (ART) model, which, like the SOM, forms clusters and is trained without supervision [Gros:86].

In the early and mid 1980s, the work by Hopfield [Hopf:82], Rumelhart and McClelland [RuMc:86], and others led to a new resurgence of the research in ANNs. The Hopfield net can be used as an associative memory (to recover stored patterns from noisy or partial information) or to solve optimization problems. Rumelhart and McClelland established the *multilayer feedforward* (MLFF) network together with its learning rule, the *backpropagation* algorithm [RuHW:86], as the major paradigm of the field. MLFF networks, also called multilayer perceptrons, overcome many of the limitations of single-layer perceptrons, and they had not been used before because an effective training algorithm for them was not available.

In the late 1980s, Lippmann provided a taxonomy of six important ANN models (the Hopfield, Hamming, and ART nets, the perceptron, the MLFF nets, and the SOMs) that can be used for classification of *static* patterns and explained the generation of decision regions by MLFF nets [Lipp:87]. He argued that ANN-based classifiers of static patterns can perform three different tasks: first, they can identify which class best represents an input pattern (a classical decision theory problem); second, they can be used as an associative memory, where the class exemplar is desired and the input pattern is used to determine which exemplar to produce; third, they can perform vector quantization or clustering of input data. According to Lippmann's taxonomy, if binary valued inputs are assumed, then the Hopfield and Hamming nets can be used with a supervised learning while the ART model allows an unsupervised training, whereas if continuous valued inputs are involved, then the perceptron and MLFF nets can be used with a supervised learning while SOMs allow an unsupervised training. Nets trained with supervision are used as classifiers or associative memories, whereas nets trained without supervision (no information concerning the correct class is provided during training) are used as vector quantizers or to form clusters.

The preceding models are not appropriate in principle for learning tasks that involve a dynamic input/output, such as sequence classification, prediction, and temporal association. Nevertheless, if some fixed-length segment of the most recent input values is considered enough to perform the task successfully, then a temporal sequence can be turned into a set of spatial patterns on the input layer of a network and MLFF nets trained by backpropagation can be used. This is, the values $x(n), x(n-1), \ldots, x(n-(m-1))$ from a (discrete) sequence $x(n)$, or the values $x(t), x(t-\Delta), \ldots, x(t-(m-1)\Delta)$ from a signal $x(t)$, are presented simultaneously at the input of the network. In a practical network, a *shift register* could be used to keep several "old" values in a buffer, or the input signal could be fed into a *delay line* that were tapped at various intervals. The resulting ANN architectures are usually called *time-delay neural networks* [HeKP:91].

In the late 1980s, time-delay nets were applied rather successfully to text pronunciation [SeRo:87], signal forecasting [LaFa:88], and speech recognition [WaHH:89]. However, there are some drawbacks to this approach: the length of the delay line or shift register must be chosen in advance, and therefore, time-delay nets do not work well with arbitrarily long sequences of relevant past inputs; if the required segment length is not short, then the large number of input units may lead to the need of a lot of training examples for successful learning and generalization, and to a slow computation in simulated nets.

Hence, also in the late 1980s and in the early 1990s, *recurrent neural network* (RNN) models including feedback connections were devised to cope naturally with the learning and computation of tasks involving sequences [Pear:89, WiZi:89, Elman:90, GiSC:90].

Some RNN models are reviewed in the excellent book by Hertz *et al.* [HeKP:91]. The key point is that the recurrency lets the network remember cues from the past and encode them in its internal state representation to accomplish with the trained task. RNN models are classified in two basic groups depending on whether they operate in *continuous-time* [Pear:89] or in *discrete-time* [Wizi:89, Elman:90].

In the latest years, different discrete-time RNN architectures have been used in some *connectionist approaches* to grammatical inference [SmZi:89, ClSM:89, Fahl:91, GiMC:92, SuGC:93, DaMo:93, DaMo:94, MaFa:94]. Most of these works have dealt with the problem of learning regular languages [CaCa:96], and just a few have addressed the problem of learning context-free languages [SuGC:93, DaMo:93]. The reported connectionist methods for grammatical inference have been based on using some network to learn either a *next-symbol prediction* task [ClSM:89] or a *string classification* task [GiMC:92]. In the former case, a stochastic presentation of positive strings is assumed, whereas in the latter case, a fixed training set containing both positive and negative examples is employed. Typically, a neural learning scheme is applied to infer a network that acts as a recognizer for some language, which may be or not the target language. Several factors must be fixed in the inference process, such as the symbol encoding, the training procedure (including the stop criterion), and the learning parameters.

If the neural network that is obtained after training is taken as the result, we may think of a connectionist GI method as an inductive inference algorithm performing a search on a hypothesis space made up of language recognizer nets. However, in some cases, a post-processing step has been added to extract from the trained network a symbolic representation of the inferred language, usually a finite-state automaton [DaDa:91, GiOm:93, ZeGS:93]. A connectionist GI method of this kind can be seen indeed as a heuristic method selecting a hypothesis in a search space of symbolic descriptions, even though a symbolic interpretation of the implicit inductive bias is not available in general.

1.6 Applications of grammatical inference

The relationship between grammatical inference and *syntactic pattern recognition* is clear: GI provides automatic methods for learning the syntactic models of the pattern classes. Unfortunately, several problems arise in practice: real patterns are often context-sensitive [Tana:95]; even in the simplest case of regular grammars, it is difficult to select an adequate solution for a particular task within a large search space of possible solutions [Micl:90]; data are normally noisy and/or incomplete, and this fact leads in some cases to the use of inefficient error-correcting techniques for the recognition step

[Tana:95]. All these causes explain why grammatical inference has been scarcely used in operational pattern recognition systems. Nevertheless, some applications have been reported in the fields of natural language modelling and translation, speech recognition, and computer vision, which are summarized in the following subsections.

1.6.1 Natural language modelling and translation

An obvious application of GI is the automated or semi-automated acquisition of *natural language* models. A wide range of grammatical and statistical formalisms, typically augmented or integrated with some kind of representation of lexical and contextual information, have been proposed in the latest decades to define plausible or, at least, useful models for natural languages like English [Allen:87, AbDa:91, Mage:94]. However, the currently available GI techniques just allow, in general terms, the learning of some simple models (e.g. n-grams) or of the basic syntactic kernel (e.g. stochastic context-free grammars) included in more complex formalisms. On the other hand, specific ad-hoc methods using partial knowledge may be applied to acquire some powerful grammatical models like unification-based grammars [OsBr:94], or a fully statistical approach may be followed to learn decision-tree based parsing models [Mage:96].

The *n-gram models* are very simple probabilistic models that have been used extensively in recent years [Jeli:91]. The n-gram models can be divided in two types: *n-gram word models* and *n-gram class models* [JaAd:94], depending on whether the conditional probabilities are estimated directly on the language words or on the classes (tags or parts-of-speech) the words belong to. In practice, n-grams are simply bigrams or trigrams, since, specially in the n-gram word case, huge information quantities and storage resources would be required to estimate and store the probabilities for larger values of n. Languages described by n-grams are equivalent to the so-called *stochastic k-testable languages in the strict sense* (*k-TS languages*), which form a very restricted subclass of stochastic regular languages that can only model local or short-term constraints [ViLl:96]. In this way, n-gram models can be inferred by applying an algorithm that learns the k-TS languages from positive examples [GaVi:90] and computing a max-likelihood estimation of the transition probabilities through data parsing [Fu:82]. Since other GI methods are available for more powerful subclasses of, and even general, regular languages (e.g. the heuristic MGGI methodology [GaVC:87,ViLl:96]), other probabilistic regular models more adequate than n-grams may be inferred from raw or tagged positive sentences.

Nevertheless, n-gram and other regular language models are still inadequate for natural language parsing models. Stochastic context-free grammars (CFGs) are more

powerful models that can be inferred from bracketed sentences using context-free GI algorithms [CrGM:78,Saka:92] and the *inside-outside* probability estimation technique [LaYo:90,Casa:94]. Note that this implies some prior knowledge of the grammar since unlabeled parse trees of the examples are needed. In some cases, the rewriting rules are acquired by some ad-hoc means and the inside-outside algorithm is applied then to assign the probabilities to each rule. Since natural language parsing models must rely heavily on lexical and contextual information to analyse sentences accurately, stochastic CFGs by themselves are neither suitable models. A few works have attempted the inference of more expressive grammatical formalisms by incorporating lexical information into the learning process [CaCh:92,OsBr:94]. However, these approaches have not achieved significant improvements over the results of stochastic CFGs on broad-coverage parsing problems [Mage:96].

If the GI field is extended to include the issue of *transduction learning*, then another related application is *natural language translation.* A transducer is a formal device which inputs strings from a given input language and outputs strings belonging to another (usually different) language. A classical hierarchy of transductions include *sequential*, *subsequential*, *finite state* or *rational* transducers, and other more complex classes such as the *pushdown* transducers. A sequential transduction is one that preserves the increasing length prefixes of input-output strings; this type of transduction is considered to be excessively restricted for most tasks of interest. On the other hand, rational transducers are quite more powerful, but no method is known so far to learn them [Vidal:94]. The subsequential transductions overcome some of the limitations of the sequential ones, and an efficient algorithm (OSTIA) is available to learn them from positive presentation of input-output examples [OnGV:93]. This algorithm has been applied to language translation tasks in limited domains as well as in other machine translation tasks [CaGV:94]. However, it is believed that natural language general translation mappings escape the capabilities of any formal device like a subsequential transducer, since a lot of semantic and contextual information is relevant to a correct translation.

1.6.2 Speech recognition

Another field of application of GI is *speech recognition* [LoRe:80, BaJM:83, Levi:85]. Speech signals constitute highly structured objects which are composed of different types of subobjects such as words, phonemes, and acoustic units. The most general problem statement is that of *continuous speech recognition*, where a sequence of utterances must be recognized from the speech data. For such a problem, a general approach includes the following five processing steps [Rabi:89]: feature analysis, speech unit matching, lexical decoding, syntactic analysis and semantic analysis.

In the *feature analysis* step, a spectral and/or temporal analysis of the speech signal is performed to provide a string of observation vectors, each containing some fixed number of features, e.g. Cepstrum coefficients. Usually, a vector quantizer is applied to map each continuous observation vector into a discrete codebook index or *microphonetic label*, which is associated with a prototype of some microphonetic category. In this way, the speech signal is translated into a string of microphonetic labels (symbols). On the other hand, some class of speech recognition units must be chosen, e.g. words or subword units such as syllables and phones. Each such unit is represented by some type of model, e.g. a *hidden Markov model* [Rabi:89], and the (spoken) instances of the speech units are represented as microphonetic label strings. The *unit matching system* provides the likelihoods of a match of the different sequences of speech recognition units to the input data. The *lexical decoding* process places constraints on the unit matching system so that the paths investigated are those corresponding to sequences of speech units that are in a word dictionary (where each word is specified in terms of the chosen speech recognition units). If the chosen units are words, then the lexical decoding step is unnecessary. Finally, the *syntactic analysis* and *semantic analysis* processes add further constraints to the set of recognition search paths, based respectively on the syntactic and semantic characteristics of the natural language that is spoken.

It is clear that the same models discussed previously for natural language modelling (n-grams, stochastic CFGs, unification-based grammars, ...) can be used in the syntactic and/or semantic analysers of a continuous speech recognition system, and hence, their inductive inference is also of interest for this application. However, for some command and control tasks, only a single word from a finite set of words is required to be recognized and there is no syntax (at the word level) or semantics to constrain the choice of words. These tasks are referred to as *isolated word speech recognition* tasks, and to solve them, only the feature analysis and speech unit matching steps are needed, whenever the word is taken as the speech recognition unit. In another class of (slightly more complicated) tasks, the *connected word recognition* tasks, a continuous utterance consisting of words from a vocabulary must be recognized, but there is no high-level syntax or semantics, e.g. to recognize a spoken string of digits.

Hidden Markov models (HMMs) and templates (these being used together with dynamic time warping alignment) have been the most extensively and successfully used models for the speech units in isolated and connected word recognition problems [RaWJ:86]. HMMs are statistical models, which are rather close to the class of stochastic finite-state automata, that can be inferred (adjusted) from training data using the *forward-backward* algorithm and the *Baum-Welch reestimation* procedure, provided that the number of states of the HMM is given in advance [Rabi:89]. An HMM is characterized by the number N of states in the model, a set of M distinct observation symbols, and three probability distributions: a state transition probability distribution

(a matrix A with $N \times N$ elements), the observation symbol probability distribution in each state (a matrix B with $N \times M$ elements), and the initial state distribution Π with N elements. Different types of HMMs have been proposed [Rabi:89]. Standard *ergodic* HMMs, in which every state can be reached in a single step from every other state, are adequate to model unstructured objects. However, this is not the case in the representation of speech units in terms of observation strings, for which more structured types of HMMs are preferred. *Left-to-right* HMMs, where no transition is allowed to a state whose index is lower than the current state, are one such type of restricted models that have been applied to speech recognition problems. In any case, once an HMM is trained for each possible word, the recognition of a spoken word is performed using *Viterbi* or *forward* algorithms [Rabi:89] to score each model based on the given sequence, and selecting the word whose model score is highest (i.e., the maximum likelihood).

It can be noted that HMMs are either little structured (ergodic HMMs) or with a predetermined rigid structure of states (e.g. left-to-right HMMs). In addition, the number of states must be known or estimated before proceeding to train an HMM. Hence, the inference of more flexible structural models of speech objects from speech data through GI techniques is interesting as an alternative to HMMs [ThGB:86, RuVi:87, GaSV:90, GaVi:90]. The resulting models are usually stochastic FSAs, either describing some subclass of stochastic regular languages [RuVi:87,GaVi:90] or the whole class of them [GaSV:90].

If the speech unit instances are represented as microphonetic label strings, two important features, that can conveniently characterize the different speech units, are the relative position of the different labels within the strings and the lengths of the substrings which represent the corresponding acoustic events. García *et al.* [GaSV:90] used this a-priori knowledge to define two morphic operators (symbol-renaming functions) for the MGGI methodology and applied them to the inference of word-modelling stochastic FSAs from samples of label strings. They reported quite good recognition results, comparable with those obtained using HMMs, in the problem of recognizing Spanish digits uttered in isolation by multiple speakers [GaSV:90]. For this isolated word recognition task, good results have been reported as well using the ECGI algorithm (an error-correcting GI technique) [RuVi:87, RuPV:89] and the GI algorithm for (stochastic) k-TS languages [GaVi:90]. Similarly to the case of using HMMs, the test strings in the recognition phase are submitted to each of the stochastic FSAs (one for each speech unit), and then they are classified following the criterion of maximum likelihood, i.e. each test string is classified in the class modelled by the stochastic FSA yielding the greatest generation probability.

1.6.3 Computer vision

A widespread belief within the community of pattern recognition researchers and practitioners is that even though syntactic pattern recognition (SPR) is a nice idea, it is unsuitable for practical applications due to the following reasons [LuVA:94]: the need to extract primitives in a robust manner; the difficulties in inferring adequate grammars or automata for the class models in the problem at hand; and the slowness of typical syntactic recognition methods like error-correcting parsing. In particular, the lack of methods to infer context-sensitive languages, together with the computational cost of parsing by context-sensitive grammars, is one of the drawbacks of the syntactic approach to pattern recognition, since in many applications the patterns of interest contain structural relationships that cannot be represented adequately by regular or context-free languages [Tana:95].

Hence, it is not surprising that the number of works reported on practical problems of pattern recognition where GI techniques have been used is small, but this is rather disappointing when compared to the conceptual importance of the subject and the amount of theoretical and methodological works on grammatical inference [Micl:90]. Nevertheless, in recent years, error-correcting regular GI approaches have been applied quite successfully to some specific problems (e.g. handwritten digit recognition [ViRV:93, LuVA:94]) showing a classification performance which is comparable or even superior to that displayed by well-established statistical pattern recognition methods. In fact, error-correcting GI techniques have incorporated statistical information as well. Furthermore, it is generally believed that by including not only statistics but also semantic information, e.g. involving the use of attributes, the performance of SPR approaches using GI may be improved.

The choice of the syntactic primitives (i.e. the terminal symbols of grammars or the automata vocabulary) is of particular concern in the case of computer vision applications. The most part of the GI literature (including the present thesis) has been devoted to the problem of learning traditional *"one-dimensional" grammars*, or equivalent automata, for which the input must be strings of symbols. Using this kind of representation, only models of object contours or profiles can be inferred in vision applications. In practice, different coding schemes have been used to translate image contours and curves into symbol strings, ranging from simple chain coding [Free:74] to attributed curve primitives [YoFu:79].

On the other hand, different types of *"multidimensional" grammars* have been proposed as more natural and powerful representations for pictorial data and object classes [BrFu:76, Fu:82, BuSa:90, Wang:92]. A multidimensional grammar is a rewriting system of the Chomsky basic form whose terminal alphabet is not composed of symbols but of more complex structures, such as trees, graphs, or 2D or 3D small

arrays (leading respectively to the types of *tree grammars*, *graph grammars*, and *array grammars*). These rewriting systems produce not strings, but trees, graphs or pictures, by assembling the primitive elements with their extended concatenation rules. Since these syntactic models can represent more complex structures than strings, their inference is a relevant problem for SPR.

However, while some GI algorithms are available for the classes of regular tree grammars and automata [Micl:90, Saka:92, RuGa:94], the problems of learning graph and array grammars seem to be quite difficult, although some attempts have been reported [Fu:82, BiDu:84, WaDa:90]. Moreover, as far as I know, and to the contrary of what might be expected, multidimensional grammars have been still less used than string grammars in actual vision applications.

In the 1970s, Professor Fu and his colleagues reported several works in which heuristic GI techniques were used within SPR approaches to some vision tasks, as summarized in the GI reviews by Fu and Miclet [Fu:82, Micl:90]. Thus, tree grammars were inferred to model image textures and fingerprint patterns, and string grammars were inferred to model shapes of object classes such as chromosomes and airplanes. Nevertheless, most of their published material was indeed at the level of feasibility studies.

More recently, two error-correcting GI algorithms for learning stochastic FSAs from (maybe noisy and/or distorted) positive examples have been employed with a high degree of success in more realistic applications. Namely, the *Markov network inference method*[2] has been applied to the classification of banded human chromosomes [ThGr:86], and the *ECGI algorithm* has been applied to both printed [ViRV:92] and handwritten digit recognition [ViRV:93, LuVA:94] (in addition to some speech recognition tasks, as mentioned previously).

The ECGI algorithm [RuVi:87] was specially designed to capture relevant regularities exhibited by the concatenation of local substructures of the considered one-dimensional patterns, as well as by the lengths of these substructures, while also obtaining appropriate models of the errors or irregularities that these patterns tend to present with respect to the learnt pattern models. Due to these characteristics, the ECGI algorithm has shown an impressive performance. For instance, using chain coded contours as input data and stochastic error-correcting (regular) parsing for recognition, the ECGI method has outperformed in correct classification rates other less structural pattern recognition approaches to handwritten digit recognition, such as the *k-nearest neighbours* technique with moment-based feature vectors [ViRV:93] and *multilayer perceptrons* trained with input vectors containing radial and Hough transform features

[2]Markov networks, as defined in [ThGr:86], constitute a subclass of stochastic FSAs.

[LuVA:94]. Likewise, the ECGI algorithm has been demonstrated to be quite superior in this task to HMMs and the MGGI methodology [LuVA:94].

1.7 Objectives of the thesis

The purpose of the present work has been to study both symbolic and connectionist learning techniques for grammatical inference and their relationships, trying to develop new GI methods and/or improvements on the existing ones. The research on GI has been restricted to the case of learning from examples, where the input consists of a finite set of members of an unknown target language (*positive sample*) and possibly a finite set of nonmembers (*negative sample*). Nevertheless, some algorithms using queries have been included in the review of the existing GI methods. Likewise, the work has been focused on the inference of non-stochastic languages, although in some cases the extension to stochastic languages can be readily accomplished, e.g. by using the learning examples to estimate probabilities once a non-stochastic representation of the language is inferred. In addition, some of the connectionist GI methods studied may be regarded as methods that implicitly infer or approximate stochastic languages.

Two classes of languages were considered as scope of the research on GI methods: the *regular languages* (RLs) and the *context-sensitive languages* (CSLs). However, since the design of GI algorithms for the whole general class of context-sensitive languages was thought to be a too ambitious aim, GI methods for some subclass of CSLs, which covered at least the descriptions of planar shapes with symmetries, were sought.

In particular, the following objectives were pursued:

- to analyse the problem of incremental GI of finite-state automata (FSA) from both positive and negative examples and to propose efficient algorithms for this problem;
- to go deeply into the study of connectionist approaches to GI in order to propose alternatives or improvements to the reported methods;
- to provide tools for a better understanding of the relationship between the symbolic and connectionist representations of finite-state machines, as well as methods for inserting, keeping and extracting symbolic information in neural networks, that allowed the design of hybrid GI techniques;
- to compare the performance of different regular GI methods, both symbolic and connectionist, on experimental benchmark tests;
- to devise some formalism that could represent and recognize a non-trivial class of CSLs, for which some practical learning method could be designed for inferring that type of description from example strings.

1.8 Structure of the thesis

The work is divided into four parts:

Part I - Introduction and review.
Part II - Contributions on symbolic and connectionist techniques for regular grammatical inference.
Part III - Augmented regular expressions and their inductive inference.
Part IV - Conclusions and future research.

The first part covers the former four chapters:

Chapter 1. Introduction.
Chapter 2. Regular grammatical inference: theory and symbolic methods.
Chapter 3. Non-regular grammatical inference through symbolic approaches.
Chapter 4. Grammatical inference through connectionist approaches.

The purpose of this first part is twofold: on one hand, to present an extensive state-of-the-art review of both symbolic and connectionist GI methods; on the other hand, to state all the basic definitions, theory, neural network architectures, notations and background that are needed to describe later my own contributions and help the reader to locate, understand and assess the performed research. The thesis contributions constitute the contents of the rest of parts (Chapters 5 to 10).

The first chapter, which is ended in this section, has been an introduction to the topic and objectives of the work, where some basic background on inductive inference, grammatical inference and applications has been given.

The review of the symbolic approaches to grammatical inference is split in two chapters. In Chapter 2, the known theory and methods for regular GI are described. In Chapter 3, the methods proposed for the inference of CFLs, CSLs, and pattern languages are reviewed.

The fourth chapter is devoted to introduce the neural network architectures, learning algorithms, and training procedures that have been used in the reported connectionist works on GI. Some experimental results of these previous works are also recalled and commented.

The second part covers Chapters 5 to 7:

Chapter 5. Regular grammatical inference from positive and negative data using unbiased finite-state automata.
Chapter 6. Regular grammatical inference using recurrent neural networks.
Chapter 7. Representation of finite-state machines in recurrent neural networks and the active grammatical inference methodology.

In this part, the contributions related to the theory and methods for regular GI are described, which include other lateral subjects such as the representation of finite-state machines in recurrent neural networks and the study of the effect of activation functions on the learning performance of these networks.

In Chapter 5, the so-called *unbiased finite-state automata* (or UFSAs) are defined and proposed as hypotheses for the problem of learning regular languages from both positive and negative examples. The theory of regular GI is reformulated according to this representation, and both non-incremental and incremental methods using UFSAs are presented which are reminiscent of the RPNI algorithm by Oncina and García [OnGa:92a-b]. A comparative empirical study of these algorithms has been carried out using a benchmark test defined by Dupont [Dupo:94]. The proposed representation and methods were reported in a communication to the SSPR'94 workshop held in Nahariya (Israel) [AlSa:95a], and part of the results of the experimental study have been included in a recent paper submitted to the *Int. Journal of Pattern Recognition and Artificial Intelligence* [AlSa:97b].

In Chapter 6, the application of recurrent neural networks (RNNs) to regular GI is discussed. Two connectionist approaches are studied: training RNNs to learn the next-symbol prediction task from a stochastic presentation of positive examples, and training RNNs to learn the string classification task from a given set of positive and negative examples. For each approach, an associated method to extract an UFSA from a trained net is proposed, which establishes a parallelism with the symbolic GI methods based on selecting a partition of the states of a canonical automaton. A preliminary version of the UFSA extraction algorithm was reported in the ICGI'94 colloquium held in Alicante (Spain) [AlSa:94b].

In addition, two experimental studies are included in Chapter 6, one for the next-symbol prediction task using the Reber grammars benchmark [ClSM:89, SmZi:89, Fahl:91], and another for the string classification task using Dupont's benchmark [Dupo:94]. Partial results of the first study were reported in [AlSa:94a,SoAl:94], and results of the second one have been partially reported in [AlSS:97] as well as included in a recently submitted paper [AlSa:97b]. The aim of the first study was to assess the influence of several types of activation functions on the learning performance of RNNs,

whereas the aim of the second one was to compare the generalization performance of first- and second-order RNNs and confront the connectionist GI methods with the symbolic GI methods tested previously on the same data (Chapter 5). Likewise, a yet unpublished theoretical discussion is included in Chapter 6 that explains the causes of the results obtained with different activation functions in RNNs.

In Chapter 7, an algebraic framework to represent finite-state machines (FSMs) in RNNs is presented that unifies and generalizes some of the previous proposals [Mins:67, GoGi:93, GoGC:94]. This theoretical work was published in the *Neural Computation* journal [AlSa:95b] and subsumes the preliminary and partial studies reported in older papers [SaAl:92,AlSa:93]. The framework is based on the formulation of both the state transition function and the output function of an FSM as a linear system of equations, and it permits an analytical explanation of the representational capabilities of first-order and higher-order single-layer RNNs. The framework, which is valid for a wide range of activation functions whenever some stability conditions are met, can be used to insert symbolic knowledge in RNNs prior to learning from examples and to keep this knowledge while training the network. Thus, a hybrid semi-automated GI methodology, called *active grammatical inference* (AGI), has been derived from this algebraic framework and reported elsewhere [SaAl:95].

The AGI methodology, which is also described in Chapter 7, combines symbolic and connectionist techniques and representations, with the novel feature that the learning process performed by an RNN can be guided by a "teacher" who introduces symbolic constraints or rules dynamically and validates partial intermediate results. The whole process is conceived as a sequence of learning cycles, each one including the steps of FSA or UFSA insertion, (possibly constrained) neural training, FSA or UFSA extraction, symbolic manipulation and validation. Although some fully-automated GI methods can be devised following the AGI methodology, e.g. by supplying learning examples incrementally in blocks (one block for each cycle) and defining a strategy to validate automatically (some part of) the extracted FSAs, the AGI approach is preferently oriented to problems where some a-priori knowledge of the solution is available.

The third part of the thesis comprises Chapters 8 and 9:

Chapter 8. Augmented regular expressions: a formalism to describe and recognize a class of context-sensitive languages.
Chapter 9. Inductive inference of augmented regular expressions.

This part deals with the so-called *augmented regular expressions* (or AREs), which are a new formalism invented by the author to represent, recognize and learn a class of CSLs, that covers pattern languages [Angl:80b], regular languages (in an obvious way), and a large subclass of CSLs capable of describing patterns with symmetries

and other context-dependent structures. An ARE is a compact description based on a regular expression, in which the star symbols are replaced by natural-valued variables (called star variables) and these variables are related through a finite number of linear equations; in this way, regular expressions (REs) are reduced to AREs with zero equations among the star variables.

In Chapter 8, AREs and their components are formally defined, and an efficient method to recognize a string as belonging or not to the language represented by an ARE is described. The recognition method is split in two stages: in the former, a string is parsed with respect to the underlying RE to yield a data structure containing the instances of the star variables for that string; in the latter, the satisfaction of the constraints included in the ARE is checked on the star instances obtained from a successful parsing by the RE. It is also demonstrated that AREs cover pattern languages [Angl:80b] but neither all CSLs nor all CFLs. The contents of Chapter 8 are basically reflected in a paper that has been published in the *Pattern Recognition* journal [AlSa:97a], in a previous technical report [AlSa:95c], and in a communication to the SSPR'96 workshop held in Leipzig (Germany) [SaAl:96].

In Chapter 9, the inductive inference of AREs from string examples is discussed. A general method to infer AREs from examples is proposed that is based on a regular GI step followed by a constraint induction process. This approach has also been reported in a communication of the ICPR'96 conference held in Vienna (Austria) [AlSa:96]. In addition, a specific method for learning AREs from positive examples has been tested, in which a connectionist regular GI step has been used. This method has been applied to the inference of a set of eight test CSLs, and the results obtained from this experimental study have been reported in the recent ICGI'96 colloquium held in Montpellier (France) [AlSC:96].

The fourth part of the thesis just includes the last chapter of the work:

Chapter 10. Conclusions and future research.

Besides enumerating the contributions of the thesis, which have been developed in Chapters 5 to 9, and summarizing the main results obtained, this last chapter points out the lines of further research that should be followed both to deepen in some of the theoretical aspects raised, mainly concerned with RNNs and AREs, and to facilitate the application of the developed GI tools to real-world problems in the fields of computer vision, speech recognition, and natural language processing.

Indeed, it must be mentioned that a pseudo-bidimensional extension of the ARE concept has already been used in a prototype system to detect and recognize traffic signs in outdoor scenes (color images of roads taken from a running car) [SaSa:96a,SaSa:96b],

and that some preliminary results of the application of the AGI methodology in learning models for the recognition of curved planar shapes have been recently reported [FoSa:97].