Grammatical Inference: An old and new paradigm

Yasubumi Sakakibara

Institute for Social Information Science, Fujitsu Laboratories Ltd., 140, Miyamoto, Numazu, Shizuoka 410-03, Japan (email: yasu@iias.flab.fujitsu.co.jp)

Abstract. In this paper, we provide a survey of recent advances in the field "grammatical inference" with a particular emphasis on the results concerning the learnability of target classes represented by deterministic finite automata, context-free grammars, hidden Markov models, stochastic context-free grammars, simple recurrent neural networks, and case-based representations.

1 Introduction

Loosely speaking, Grammatical Inference is an inductive inference problem where the target domain is a formal language and the representation class is a family of grammars. The learning task is to identify a "correct" grammar for the (unknown) target language, given a finite number of examples of the language. Grammatical Inference is a well-established research field in Artificial Intelligence as it dates back to the 60s. Gold [25] originated this study and introduced the notion of *identification in the limit*. His motivation for studying the problem is to construct a formal model of human language acquisition. Since his seminal work, there has been a remarkable amount of work to establish a theory of grammatical inference, to find effective and efficient methods for inferring grammars, and to apply those methods to practical problems.

Grammatical inference has been investigated, more or less independently, within many research fields, including machine learning, computational learning theory, pattern recognition, computational linguistics, neural networks, formal language theory, information theory, and many others. Recently, the international conference on Grammatical Inference has been established with an aim to bring together researchers from diverse fields and to bring about a stimulating interdisciplinary interaction between them. The first colloquium on grammatical inference was held in U.K. in April 1993, and the second one in Spain in September 1994 [19].

There are several excellent survey articles on the field of grammatical inference. An early survey on inductive inference is Angluin and Smith's article [14]. An early good introduction to grammatical inference is Miclet's article [38]. A recent extensive survey of the inference of deterministic finite automata is Pitt's paper [42].

Much of the recent research activities on grammatical inference have been stimulated by the new learning models proposed recently within computational learning theory framework: the query learning model of Angluin [10] and the PAC (probably approximately correct) learning model of Valiant [59]. These new models put much more emphasis on the computational efficiency of the inference algorithm. A good introduction to computational learning theory is Laird's paper [34], and a very recent survey of computational learning theory is Angluin's paper [12]. Thus grammatical inference is an old and new paradigm in artificial intelligence.

This paper, rather than being a thorough survey on the topic, is intended mainly as a review of the research carried out by the Machine Learning group at Fujitsu Laboratories Ltd. and related work done at other institutions. The interested reader can consult the above cited survey papers. Also, an important subject which we won't deal with here is inductive inference of *indexable classes* of formal languages (e.g., see [14] and [63] for references). This is because, while we acknowledge that enumeration is a powerful and useful technique in inductive inference, we are more interested in "constructive" methods in the sense that the grammars are constructed directly from training examples rather than by enumeration.

We will begin with the problem of identifying deterministic finite automata (DFAs) from examples. DFAs are the bottom class of formal grammars in the Chomsky hierarchy, and the problem of identifying DFAs from examples has been studied quite extensively [14, 42]. We will pick up several interesting results on identifying DFAs: polynomial-time identification of DFAs from queries, identification of subclasses of DFAs from positive data, computationally hardness results, and identification from erroneous examples. In Section 4, we will consider the problem of identifying context-free grammars (CFGs) because the questions of whether there are analogous results held for context-free grammars would be more interesting and important. The results contain identification of CFGs from examples in the form of structured strings, polynomial-time reduction to identification of finite automata, and efficient identifications of several subclasses of CFGs. In Section 5, since stochastic modeling is very important for practical applications, we will consider the problem of identifying stochastic grammars. A stochastic grammar is obtained by specifying a probability for each production in a grammar. We will review some fundamental methods for training probabilistic parameters in the grammar based on expectation maximization (EM), and their applications to biological sequence analyses. In Section 6, we will see two special topics which use non-grammatical representations for grammatical inference or language learning. One is simple recurrent neural networks and the other is case-based representations.

2 The Learning Models

Within computational learning theory, there are three major established formal models for learning from examples or inductive inference: the *identification in* *the limit* by Gold [25], the *query learning model* by Angluin [10], and the *PAC learning model* by Valiant [59]. Each model provides a learning protocol and a criterion for the success of learning. Identification in the limit views learning as an infinite process and provides a learning model where an infinite sequence of examples of the unknown grammar G is presented to the inference algorithm M and the eventual or limiting behavior of the algorithm is used as the criterion of its success. A *complete presentation* of the unknown grammar G is an infinite sequence of ordered pairs $\langle w, l \rangle$ from $\Sigma^* \times \{0,1\}$ such that $l = 1$ if and only if w is generated by G, and such that every string w of Σ^* appears at least once as the first component of some pair in the sequence, where Σ is the terminal alphabet. If after some finite number of steps in a complete presentation of G, M guesses a correct grammar which is equivalent to the unknown grammar G and never changes its guess after this, then M is said to *identify G in the limit from complete presentations.*

Angluin [10] has considered a learning situation in which a teacher is available to answer specific kind of queries on the unknown grammar G and devised an elegant formulation of such a teacher and learner paradigm. In this setup, we can expect the inference algorithm be the *exact identification*, which means the algorithm outputs a correct grammar in a certain finite time. This is no longer a limiting criterion of learning. In the query learning model, a teacher is a fixed set of *oracles* that can answer specific kinds of queries made by the inference algorithm on the unknown grammar G. For example, the following two types of queries are typical:

1. *Membership*. The input is a string $w \in \Sigma^*$ and the output is "yes" if w is generated by G and "no" otherwise.
2. *Equivalence*. The input is a grammar G' and the output is "yes" if G' is equivalent to G (i.e., G' generates the same language as G) and "no" otherwise. If the answer is "no", a string w in the symmetric difference of the language generated by G and the language generated by G' is returned.

For the equivalence query, the returned string w is called a *counter-example*. A membership query returns one bit of information. Nevertheless it often plays an important role in efficient exact identification.

Valiant [59] has introduced the distribution-independent probabilistic model of learning from random examples, which is called *probably approximately correct learning* (PAC learning, for short). In the PAC learning model, we assume that random samples are drawn independently from the domain Σ^* whose probability distribution D may be arbitrary and unknown. The inference algorithm takes a sample as input and produces a grammar as output. The success of identification is measured by two parameters: the accuracy parameter ϵ and the confidence parameter δ, which are given as inputs to the inference algorithm. A successful inference algorithm is one that *with high probability* (at least $1 - \delta$) finds a grammar *whose error is small* (less than ϵ).

We measure the efficiency of the inference algorithm with respect to relevant parameters: the size of examples and the size of the unknown grammar. The *Size* of an example in the form of string is the length of the string. The *Size* of the unknown grammar is usually the number of states, in the case of finite automata, and the number of production rules, in the case of context-free grammars.

3 Learning Finite Automata

The study of the identifiability of deterministic finite automata is an excellent mean for studying a number of general aspects of inductive inference and grammatical inference [42]. In this section, we will review several important results and useful techniques related to computationally efficient identifications of deterministic finite automata.

A *deterministic finite (state) automaton* (DFA) is defined by a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is an alphabet of input symbols, δ is the *state-transition function* $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is a set of *final states.* The *language accepted* by a DFA A is denoted by $L(A)$.

3.1 Learning from representative samples

When trying to identify an unknown DFA $A = (Q, \Sigma, \delta, q_0, F)$ from examples, a useful information about A is the *representative sample* S of A, that is, a finite subset of $L(A)$ that exercises every live transition in A. Taking the set $R(S)$ of all prefixes of strings in S, for every live state q of A, there must exist a string u in $R(S)$ such that $\delta(q_0, u) = q$. Further, for every state q and every transition $\delta(q, a)$ from q where $a \in \Sigma$, there exists a string va in $R(S)$ such that $\delta(q_0, v) = q$ and $\delta(q, a) = \delta(q_0, va) = q'$. Thus every state and transition are represented by strings in $R(S)$. It remains to distinguish two states q_u and q_v represented by two strings u and v in $R(S)$, i.e., $q_u = \delta(q_0, u)$ and $q_v = \delta(q_0, v)$, if q_u and q_v are different states in A. Angluin [7] has given an efficient procedure to solve this problem using membership queries. A *membership query* made by an inference algorithm proposes a string w and asks whether $w \in L(A)$, where A is the unknown DFA. The answer is either “yes” or “no”.

Theorem 1 [7]. *The class of deterministic finite automata can be identified in polynomial time from a representative sample and using membership queries.*

3.2 Learning with teachers

Angluin [9] has considered a learning protocol which is based on what is called “minimally adequate teacher”. This teacher can answer two types of queries about the unknown DFA A made by an inference algorithm: *membership query* and *equivalence query.* An *equivalence query* proposes a conjecture A' of DFA and asks whether $L(A) = L(A')$. The answer is either “yes” or “no”. If it is “no”, then it provides a *counterexample,* an arbitrary string w in the symmetric difference of $L(A)$ and $L(A')$. Angluin [9] has shown that the equivalence query compensates for the lack of representative samples, and presented an efficient inference algorithm for identifying DFAs using equivalence and membership queries.

Theorem 2 [9]. *The class of deterministic finite automata can be identified in polynomial time using equivalence queries and membership queries.*

Yokomori [61] has studied efficient identification of *non-deterministic* finite automata from equivalence and membership queries.

3.3 Learning from positive data

One interesting and important topic on the Gold's framework of identification in the limit for language learning is identification from positive data. A *positive presentation* of the unknown DFA A is any infinite sequence of examples such that the sequence contains all and only the strings in the language $L(A)$. Gold [25] has shown that there is a fundamental, important difference in what could be learned from *positive* versus *complete* presentations, and shown a negative result that no "superfinite" class of languages can be identified in the limit from positive presentation. Since the class of regular languages is superfinite, we need to restrict DFAs somehow to subclasses to establish identifiability results from positive presentation.

The problem is to avoid "overgeneralization", which means guessing a language that is a strict superset of the unknown language. Angluin [8] has introduced a series of subclasses of DFAs, called k-reversible automata for $k = 0, 1, 2, \ldots$, and shown that the existence of characteristic samples is sufficient for identification from positive presentation (to avoid overgeneralization) for k-reversible automata and there exist such characteristic samples for the class of k-reversible automata. A *characteristic sample* of a k-reversible automaton A is a finite sample $S \subset L(A)$ such that $L(A)$ is the smallest k-reversible language that contains S. It turns out that any characteristic sample is a representative sample for k-reversible automata.

As we have seen in Section 3.1, a representative sample provides enough information for reconstructions of states and state transitions. By utilizing the structural properties specific to k-reversible automata, we could accomplish the main task of state distinctions in identifying k-reversible automata without the use of membership queries. For example, a *zero-reversible automaton* is a DFA such that it has at most one final state and no two edges entering any state are labeled with the same symbol. Given a representative sample S for the unknown zero-reversible automaton, we construct the prefix tree automaton A' that precisely accepts the set S, and then merge states in A' to satisfy the conditions for zero-reversible automata.

Theorem 3 [8]. *The class of k-reversible automata, for $k = 0, 1, 2, \ldots$, can be identified in the limit from positive presentation.*

Furthermore, the inference algorithm updates a conjecture in time polynomial in the size of the inputs.

Another interesting class of DFAs which can be identified in the limit from positive presentation is the class of strictly deterministic automata investigated by Yokomori [62]. A *strictly deterministic automaton* is a DFA such that the set of labels W for state-transition edges is extended to be a finite subset of strings over Σ, each edge has the unique label (no same label is attached to different edges), and for each symbol $a \in \Sigma$ there is at most one label in W starting with a.

Theorem 4 [62]. *The class of strictly deterministic automata can be identified in the limit from positive presentation.*

An inference algorithm can be constructed so that it not only runs in time polynomial in m, the maximum length of all positive examples provided, but also makes at most a polynomial number of *implicit errors of prediction* in m and n, the size of the unknown strictly deterministic automaton.

Other interesting topics and results on identification from positive presentation which may not directly be related to DFAs are Angluin's characterization of identifiability from positive presentation [6], Angluin's *pattern languages* [5], Koshiba's extension to *typed pattern languages* [32], Shinohara's general result for identifiability from positive presentation [55], and Oncina et al.'s *subsequential transducers* [40].

3.4 Hardness results

There are many computationally hardness results related to identifying DFAs. Gold [26] has shown that the problem of finding a DFA with a minimum number of states consistent with a given finite sample of positive and negative examples is NP-hard. This result is generally interpreted as indicating that even a very simple case of grammatical inference, identifying DFA from positive and negative examples, is computationally intractable. Further, Pitt and Warmuth [43] have proven a stronger result, namely that it is NP-hard to find a DFA of at most $n^{(1-\epsilon)\log\log n}$ states consistent with a given finite sample of positive and negative examples for any constant $\epsilon > 0$, where n is the number of states of a minimum DFA consistent with the given sample.

Angluin [11] has shown negative results for efficient identifications of various classes of grammars from equivalence queries only. She has developed the useful technique of "approximate fingerprints" to obtain negative results for identification from equivalence queries only.

3.5 Learning from erroneous examples

In practice, it is natural to assume that the examples may contain some noise. There are fewer works to study the effect of noise on learning from queries in the Valiant's probabilistic framework of PAC-learnability.

Sakakibara [48] has defined a benign model for errors in the responses to membership queries where answers to queries are subject to random independent noise (i.e., for each query there is some independent probability to receive an incorrect answer and these errors are not persistent), and shown that these errors can be effectively removed by repeating the query until the confidence in the correct answer is high enough.

Ron and Rubinfeld [46] have considered a model of *persistent* noise in membership queries in which a fixed but randomly chosen fraction of membership queries are answered incorrectly but any additional query on the same string is answered consistently when queried again. They have shown by modifying Angluin's algorithm (Theorem 2) for identifying DFAs using equivalence and membership queries that DFAs can be learned in polynomial time from membership queries with persistent noise under the uniform distribution on inputs.

Sakakibara and Siromoney [53] have studied a noise model which is specific to language learning where the examples are corrupted by purely random errors affecting only the strings (and not the labels). They have considered three types of errors on strings, called *EDIT operation errors.* EDIT operations consist of "insertion", "deletion", and "change" of a symbol in a string. They have shown efficient identification from random examples with EDIT noise for a small subclass of regular languages defined by containment decision lists, a variant of *decision list* [45] to represent languages.

4 Learning Context-Free Grammars

As we have seen in the previous sections, there has been extensive research into the problem of identifying DFAs from examples. The question of whether there are analogous results for context-free grammars is important because context-free grammars are a more interesting class of grammars from the practical point of view.

A *context-free grammar* (CFG) is defined by a quadruple $G = (N, \Sigma, P, S)$, where N is an alphabet of *nonterminal symbols*, Σ is an alphabet of *terminal symbols* such that $N \cap \Sigma = \emptyset$, P is a finite set of production rules of the form $A \rightarrow \alpha$ for $A \in N$ and $\alpha \in (N \cup \Sigma)^*$, and S is a special nonterminal called the *start symbol*. The *language generated* by a CFG G is denoted $L(G)$.

Angluin [11] has shown that the whole class of CFGs cannot be identified in polynomial time using equivalence queries only. Furthermore, Angluin and Kharitonov [13] have shown that the problem of identifying the class of CFGs from membership and equivalence queries is computationally as hard as the cryptographic problems for which there is currently no known polynomial-time algorithm. Despite these negative results, we will present in the following sections several positive results for identifying the whole class of CFGs with additional information or identifying subclasses of CFGs efficiently.

4.1 Learning from structural information

We consider an identification problem for CFGs where, besides given examples, some additional information is available for the inference algorithm. A useful (and maybe reasonable) information would be information on the grammatical structure of the unknown CFG. We assume example presentations in the form

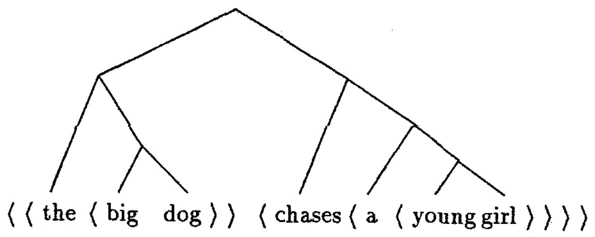


Fig. 1. An example of structured string for "the big dog chases a young girl".

of strings with grammatical structure. Levy and Joshi [36] have already suggested the possibility of efficient grammatical inferences in terms of strings with grammatical structure.

A string with grammatical structure, called *a structured string* or *a structural description* (of string), is a string with some parentheses inserted to indicate the shape of the derivation tree of a CFG, or equivalently an unlabeled derivation tree of the CFG, that is, a derivation tree whose internal nodes have no labels. (See Figure 1.) It is known that the set of derivation trees of a CFG constitutes a rational set of trees, where a *rational set* of trees is a set of trees which can be recognized by some tree automaton. Further, the set of unlabeled derivation trees of a CFG also constitutes a rational set of trees. Based on these observations, the problem of identifying CFGs from structured strings is reduced to the problem of identifying tree automata.

Sakakibara [47] has shown by extending Angluin's inference algorithm (Theorem 2) for DFAs to tree automata that the class of CFGs can be identified in polynomial time using structural membership queries and structural equivalence queries.

Theorem 5 [47]. *The class of context-free grammars can be identified in polynomial time using structural equivalence queries and structural membership queries.*

A *structural membership query* is a membership query for a structured string, and a *structural equivalence query* returns "yes" if a queried CFG is *structurally equivalent* to the unknown CFG and returns "no" with a counterexample otherwise.

Since the class of CFGs is superfinite, Gold's negative result [25] on identifiability from positive presentation implies that the class of CFGs cannot be identified in the limit from positive presentation. Sakakibara [49] has demonstrated that, here also, information on the grammatical structure of the unknown CFG could help the inference. He has shown that there exists a class of CFGs, called *reversible context-free grammars*, which can be identified in the limit from positive presentations of structured strings, that is, all and only unlabeled derivation trees of the unknown CFG, and shown that the reversible context-free grammar is a normal form for CFGs, that is, reversible context-free grammars can generate all the context-free languages.

A *reversible context-free grammars* is a CFG $G = (N, \Sigma, P, S)$ such that $A \rightarrow \alpha$ and $B \rightarrow \alpha$ in P implies that $A = B$ and $A \rightarrow \alpha B \beta$ and $A \rightarrow \alpha C \beta$ in P implies that $B = C$, where A, B, and C are nonterminals, and $\alpha, \beta \in (N \cup \Sigma)^*$.

Theorem 6 [49]. *The class of reversible context-free grammars can be identified in the limit from positive presentation of structured strings.*

Since the inference algorithm for reversible context-free grammars is an extension of Angluin's inference algorithm which identifies zero-reversible automata (Theorem 3), the algorithm updates a conjecture in time polynomial in the size of the inputs. Note that the above result does not imply that the whole class of CFGs can be identified from positive presentation of structured strings.

A related early work to identifying CFGs from positive presentation of structured strings is Crespi-Reghizzi's [20]. He has described a constructive method for identifying a subclass of CFGs, which is a different class from reversible CFGs, from positive samples of structured strings. His class of CFGs defines only a subclass of context-free languages, called *noncounting context-free languages.* Mäkinen [37] has refined Sakakibara's inference algorithm for reversible CFGs to gain more efficiency, and also investigated a subclass of reversible CFGs, called *type invertible grammars*, that can be identified from positive presentation of structured strings in time linear in the size of the inputs.

4.2 Reductions to finite-automata learning problems

A well-known technique often used to establish identifiability results is a *reduction* technique that reduces an inference problem to an other inference problem whose result is known. Takada [57] has shown that the inference problem for even linear grammars can be solved by reducing it to the one for DFAs, and presented a polynomial-time algorithm for the reduction. For example, we can identify the class of even linear grammars using equivalence and membership queries in polynomial time by employing Angluin's efficient algorithm for DFAs (Theorem 2) via reduction.

An *even linear grammar* is a CFG that has productions only of the form $A \rightarrow uBv$ or $A \rightarrow w$ such that u and v have the same length, where A and B are nonterminals and u, v and w are strings over Σ. Let $G = (N, \Sigma, P, S)$ be an even linear grammar. We write $x \overset{\pi}{\Longrightarrow} y$ to mean that y is derived from x applying the production π in P, where $x, y \in (N \cup \Sigma)^*$. We denote a derivation from x_0 to x_k obtained by applying a sequence $\gamma = \pi_1 \pi_2 \cdots \pi_k$ of productions by $x_0 \overset{\gamma}{\Longrightarrow} x_n$. γ is called an *associate word* and a set of associate words is called a *control set* on G. The language generated by G with a control set C is defined by $L(G, C) = \{w \in \Sigma^* \mid S \overset{\gamma}{\Longrightarrow} w \text{ and } \gamma \in C\}$. It can be shown that there is a universal even linear grammar G_U such that for any even linear grammar G, $L(G) = L(G_U, C)$ for some regular control set C.

Theorem 7 [57]. *The problem of identifying the class of even linear grammars is reduced to the problem of identifying the class of finite automata.*

Note that the class of even linear languages properly contains the class of regular languages and is a proper subclass of context-free languages. By iteratively applying the above reduction technique, Takada [58] has further developed an infinite hierarchy of families of languages whose identification problems are reduced to the identification problem of DFAs.

4.3 Learning subclasses of context-free grammars

Because the whole class of CFGs seems to be hard to be identified efficiently without any additional information, there have been some attempts to design polynomial-time algorithms for identifying *subclasses* of CFGs from examples.

Ishizaka [30] has investigated a subclass of CFGs, called *simple deterministic grammars*, and gave a polynomial-time algorithm for exactly identifying it using equivalence and membership queries in terms of general CFGs. This inference algorithm may sometimes ask an equivalence query for a CFG which is not simple deterministic.

A CFG $G = (N, \Sigma, P, S)$ in 2-standard form is called *simple deterministic* if $A \rightarrow a\alpha$ and $A \rightarrow a\beta$ in P implies that $\alpha = \beta$, where A and B are nonterminals, a is a terminal, and $\alpha, \beta \in (N \cup \Sigma)^*$.

Theorem 8 [30]. *The class of simple deterministic grammars can be identified in polynomial time using equivalence queries and membership queries in terms of general context-free grammars.*

Note that given any regular language L, the language $L\#$ is simple deterministic, where $\#$ is a special symbol not in Σ. In this sense, the class of simple deterministic languages properly contains the class of regular languages.

Yokomori [60] has considered a smaller class of simple deterministic grammars with the goal of finding a polynomial-time algorithm to identify it in the limit from positive presentation. A CFG $G = (N, \Sigma, P, S)$ in Greibach normal form is called *very simple* if for each terminal symbol a in Σ, there exists exactly one production rule starting with a (i.e., exactly one production rule of the form $A \rightarrow a\alpha$, where $\alpha \in (N \cup \Sigma)^*$). He has shown that the class of very simple grammars can efficiently be identified in the limit from positive presentation, and this result has provided the first instance of language class containing non-regular languages that can be identified in the limit in polynomial time in the sense of Pitt [42], that is, the time for updating a conjecture is bounded by a polynomial in the size n of the unknown grammar and the sum of lengths of examples provided, and the number of times the inference algorithm makes a wrong conjecture is bounded by a polynomial in n.

Theorem 9 [60]. *The class of very simple grammars can be identified in the limit from positive presentation in polynomial time.*

From this result, it immediately follows that the class of very simple grammars can be identified in polynomial time using only equivalence queries.

Related to identification of very simple grammars, Burago [18] has investigated the structurally reversible context-free grammars, and shown that the class of structurally reversible CFGs can be identified in polynomial time using equivalence queries and membership queries. A CFG is called *structurally reversible* if among all nonterminal strings that might derive a given terminal string, no one is an extension of the other. The class of structurally reversible CFGs is a subclass of CFGs and the class of structurally reversible context-free languages properly contains the class of very simple languages.

Other representation forms for languages which are not in the form of grammars sometimes help understanding the mathematical structures and designing efficient inference algorithms. Fahmy and Biermann [23] have investigated identification of *real time acceptors*. The class of languages accepted by real time acceptors is a subclass of context-sensitive languages and incomparable with the class of context-free languages.

5 Learning Stochastic Grammars

Another major research topic in grammatical inference is stochastic modeling and training of stochastic grammars. Stochastic modeling has become increasingly important for applications such as speech recognition, natural language processing, and biological sequence analysis. A *stochastic grammar* is obtained by specifying a probability for each production in a grammar. A stochastic grammar assigns a probability to each string which it derives and hence defines a probability distribution on the set of strings. Stochastic (probabilistic) automata are the probabilistic counterpart of finite automata that are known as *hidden Markov models* (HMMs) and very extensively used in speech recognition. *Stochastic context-free grammars* (SCFGs) is a superclass of and goes one step beyond hidden Markov models in the Chomsky hierarchy.

The problem of identifying stochastic grammars has two aspects: determining the discrete structure (topology) of the grammar and estimating probabilistic parameters in the grammar. Based on the maximum likelihood criterion, efficient estimation algorithms for probabilistic parameters have been proposed: *forward-backward algorithm* for HMMs [44] and *inside-outside algorithm* for SCFGs [16, 35]. The relative success of stochastic grammars in real tasks is due to the existence of these techniques for automatic estimation of probabilities and distributions. Both algorithms are iterative algorithms which are based on the expectation-maximization (EM) technique that increases the likelihood of the training sample in each step until a local maximum is reached. Therefore, the initialization in the iterative process is a crucial point since it affects the speed of convergence and the goodness of the results. On the other hand, finding an appropriate discrete structure of the grammar is a harder problem. In certain cases, it might be possible to consider the inference of the discrete structure as a result of the probability estimation process. For example, in the case of HMM, we start with a fully connected HMM, get a locally maximum estimation of probabilities, and obtain a structure of HMM by pruning out zero or low probability transitions. However, this method does not seem to be effective or efficient. In fact, Abe and Warmuth [2] have shown a computationally hardness result for the inference of probabilistic automata.

In the remaining of this section, we will focus on probability estimation procedures for HMM and SCFG.

5.1 Hidden Markov models

A *Hidden Markov Model* (HMM) is defined by a 5-tuple $\lambda = (Q, \Sigma, T, O, \pi)$, where Q is a finite set of states, Σ is an alphabet of output symbols, T is a state transition probability distribution, O is an output symbol probability distribution, and π is an initial state distribution. Let $Q = \{q_1, \ldots, q_n\}$. T is the set $\{t_{ij} \mid 1 \leq i, j \leq n\}$ of state transition probabilities where t_{ij} is a state transition probability from state q_i to state q_j, O is the set $\{o_j(a) \mid 1 \leq j \leq n, a \in \Sigma\}$ of output symbol probabilities where $o_j(a)$ is a probability to output a at state q_j, and π is the set $\{\pi_i \mid 1 \leq i \leq n\}$ of initial state probabilities where π_i is the probability to start at state q_i.

Given a HMM λ, there are three basic problems for dealing with λ: given a string $w = a_1 \cdots a_m$,

1. calculate $\Pr(w|\lambda)$, the probability of the string w,
2. find the most probable path $s = q_{i_1} \cdots q_{i_l}$ of states to maximize $\Pr(s|w, \lambda)$,
3. estimate the parameters in λ to maximize $\Pr(w|\lambda)$.

These problems can be solved efficiently using dynamic programming techniques [44]. A polynomial-time algorithm for solving the second algorithm is known as *Viterbi* algorithm, and a polynomial-time algorithm for the third problem is known as *Forward-Backward (Baum-Welch)* algorithm. To solve the first problem, we consider the forward variable $\alpha_k(q_i)$ defined as $\alpha_k(q_i) = \Pr(a_1 \cdots a_k, q_i|\lambda)$, i.e., the probability of the initial segment $a_1 \cdots a_k$ of the string w and state q_i at time k. The probability $\alpha_k(q_i)$ can be calculated inductively as follows:

1. Initialization:
$$\alpha_1(q_i) = \pi_i o_i(a_1)$$
2. Induction:
$$\alpha_{k+1}(q_j) = \left(\sum_{i=1}^{N} \alpha_k(q_i) t_{ij}\right) o_j(a_{k+1})$$
3. Termination:
$$\Pr(w|\lambda) = \sum_{i=1}^{N} \alpha_m(q_i).$$

The forward-backward algorithm is an EM (expectation maximization) algorithm which finds parameters in the HMM λ to maximize $\Pr(w|\lambda)$. It proceeds as follows:

1. Let λ_{old} be an initial guess for the parameters.
2. Based on λ_{old} and the given string w,
   (a) For each pair q_i, q_j of states, estimate the fraction of times a transition is made from q_i to q_j among all transitions out of q_i. In λ_{new}, set t_{ij} to this value.
   (b) For each state q_j and output symbol a, estimate the fraction of times that a is output in state q_j. In λ_{new}, set $o_j(a)$ to this value.
3. Set $\lambda_{old} = \lambda_{new}$ and iterate starting at step 2 until there are no significant changes in λ_{old}.

The forward-backward algorithm for HMMs is very efficient because of the use of dynamic programming techniques, including the forward procedure and the symmetric "backward" procedure. Each iteration in the algorithm increases $\Pr(w|\lambda)$, but the algorithm can still get caught in local maxima. The algorithm is easily extended to handle a set of strings, but the algorithm suffers from the usual problems with maximum likelihood estimates: when it observe something 0 times, it sets the probability to 0.

5.2 Stochastic context-free grammars

A *stochastic context-free grammar* (SCFG) G consists of a set of nonterminal symbols N, a terminal alphabet Σ, a set P of production rules with associated probabilities, and the start symbol S. The associated probability for every production $A \rightarrow \alpha$ in P is denoted $\Pr(A \rightarrow \alpha)$, and a probability distribution exists over the set of productions which have the same nonterminal on the left-hand sides.

The three basic problems to deal with SCFGs which are same as in HMMs can be solved efficiently. The first two problems, calculating the probability $\Pr(w|G)$ of a given string w assigned by a SCFG G and finding the most likely derivation tree of w by G, can be solved using dynamic programming methods analogous to the Cocke-Kasami-Young or Early parsing methods [4]. There is a standard method for estimating the parameters of an SCFG (i.e. the probabilities of the productions) from a set of training strings. This procedure is known as the *inside-outside* algorithm [35]. Just like the forward-backward algorithm for HMMs, this procedure is an expectation-maximization (EM) method for obtaining maximum likelihood of the grammar's parameters. However, it requires the grammar to be in Chomsky normal form, which is inconvenient to handle in many practical problems (and requires more nonterminals). Further, it takes time at least proportional to n^3, whereas the forward-backward procedure for HMMs takes time proportional to n^2, where n is the length of the typical string. There are also many local maxima in which the method can get caught.

To avoid such problems, Sakakibara et al. [50] have developed a new method for training SCFGs that is a generalization of the forward-backward algorithm to tree grammars and which is more efficient than the inside-outside algorithm. The new algorithm, called Tree-Grammar EM, requires structured strings as training examples. This algorithm uses a similar idea to identification of CFGs from structured strings shown in Section 4.1. Since information on the grammatical structure is given explicitly in training strings, Tree-Grammar EM does not have to (implicitly) consider *all* possible derivations of the training strings when reestimating the grammar's parameters, as the inside-outside algorithm must do. This reduces the time complexity to a time proportional to n per training string of length n, and hence may be practical on longer strings. Tree-Grammar EM also tends to converge faster because each training structured string is much more informative.

Sakakibara et al. [50] have also modified the algorithm to train SCFGs even from (unstructured) strings. If only unstructured training strings are available, we iteratively estimate the structure of the training strings as follows:

1. Start with a initial grammar and parse the training strings to obtain a set of partially structured strings.
2. Estimate a new SCFG using the partially structured strings and the estimation algorithm Tree-Grammar EM.
3. Use the trained grammar to obtain more accurately structured training strings.
4. Repeat steps 2 and 3 until finding the structures stabilizes.

In natural language processing, Pereira and Schabes [41] have developed a similar method to Tree-Grammar EM for training SCFGs from bracketed sentences to incorporate linguistic information. Their method utilizes phrase bracketing information during the estimation process of the inside-outside algorithm to get a linguistically-motivated maximum.

Stolcke and Omohundro [56] have considered identification of a discrete structure of the stochastic grammar. They have proposed efficient heuristic methods for finding the topology of HMM and for finding an appropriate set of production rules of SCFG based on Bayesian criterion, and shown some experimental results.

5.3 Applications to molecular sequence analyses

Attempts to understand the folding, structure, function and evolution of molecules have resulted in the confluence of many diverse disciplines ranging from structural biology and chemistry, to computer science and computational linguistics. Rapid generation of sequence data in recent years thus provides abundant opportunities for developing new approaches, to problems in computational biology [29]. Determining common or consensus patterns among a family of sequences, producing a multiple sequence alignment, discriminating members of the family from non-members and discovering new members of the family will continue to be some of the most important and fundamental tasks in mathematical analysis and comparison of macromolecular sequences.

Recently, Searls [54, 29] has argued the benefits of viewing the biological strings representing DNA, RNA and protein as sentences derived from a formal grammar. In this new direction of computational biology research, stochastic context-free grammars have been applied to the problems of folding, aligning and modeling families of tRNA sequences [50]. SCFGs capture the sequences' common primary and secondary structure (See Figure 2) and generalize the HMMs used in related work on protein and DNA. Results show that after having been trained on as few as 20 tRNA sequences from only two tRNA subfamilies (mitochondrial and cytoplasmic), the model can discern general tRNA from similar-length RNA sequences of other kinds, can find secondary structure of new tRNA sequences, and can produce multiple alignments of large sets of tRNA sequences. Figure 3 shows an example of multiple sequence alignment (which is a central problem in computational biology) produced by the learned grammar: the learned grammar has successfully produced a very accurate multiple alignments for some family (gene) of molecular sequences, called tRNA.

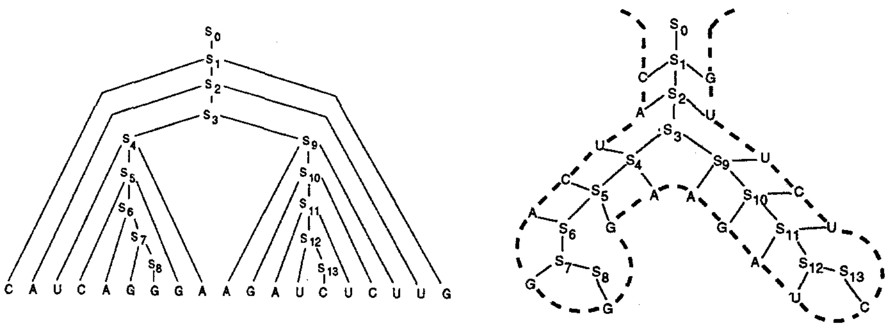


Fig. 2. A derivation tree (left) generated by a simple CFG for RNA molecules and the physical secondary structure (right) of the RNA sequence which is a reflection of the derivation tree.

Related to the above work, Krogh et al. [33] have applied HMMs to the problems of statistical modeling, database searching and multiple sequence alignment of protein families and protein domains. These methods are demonstrated on the globin family, the protein kinase catalytic domain, and the EF-hand calcium binding motif. In each case, the parameters of an HMM are estimated from a training set of unaligned sequences. The HMM produces multiple alignments of good quality that agree closely with the alignments produced by programs that incorporate three-dimensional structural information. When employed in discrimination tests, the HMM is able to distinguish members of these families from non-members with a high degree of accuracy.

Recently, Abe and Mamitsuka [1] have studied a more powerful class of grammars, called *stochastic ranked node rewriting grammars*, than SCFGs and applied it to the problem of secondary structure prediction of proteins.

```
    <      D-domain      > <    Anticodon    >< Extra ><     T-domain    >
((((  ((((             )))) (((((  ===  )))))          (((((       )))))))
AAGGUGGCAGAGUUCGGCCUAACGCGGCGGCCUGCAGAGCCGCUC----AUCGCCGGUUCAAAUCCGGCCCU
CGUGUGGCGUAGUC-GGU--AGCGCGCUCCCUUAGCAUGGGAGAG----GUCUCCGGUUCGAUUCCGGACUC
CCCAUCGUCUAGA--GGCCUAGGACACCUCCCUUUCACGGAGGCG----A-CGGGGAUUCGAAUUCCCCUGG
GGCAUAGCCAAGC--GGU--AAGGCCGUGGAUUGCAAAUCCUCUA----UUCCCCAGUUCAAAUCUGGGUGC
UUUGUAGUUUAUGUG-----AAAAUGCUUGUUUGUGAUAUGAGUGAAAU-------------------UGG

((((  ((((              )))) (((((  ===  )))))          (((((       )))))))
AAGGUGGCAG.AGUUcGGccUAACGCGGCGGCCUGCAGAGCCGCUC---AUCGCCGGUUCAAAUCCGGCCCU
CGUGUGGCGU.AGUC.GG..UAGCGCGCUCCCUUAGCAUGGGAGAGG---UCUCCGGUUCGAUUCCGGACUC
CC-AUCGUCU.AGAG.GCc.UAGGACACCUCCCUUUCACGGAGGCG----ACGGGGAUUCGAAUUCCCCU-G
GGCAUAGCCA.AGC-.GG..UAAGGCCGUGGAUUGCAAAUCCUCUA---UUCCCCAGUUCAAAUCUGGGUGC
UUUGUAGUUU.A--U.GU..GAAAAUGCUUGUUUGUGAUAUGAGUGA--AAU-----------------UGG
```

Fig. 3. Comparison of the alignment of several representative tRNAs produced by trained (learned) grammar (bottom) with that from the biologically trusted database (top). Parentheses indicate base-paired positions; === the anticodon.

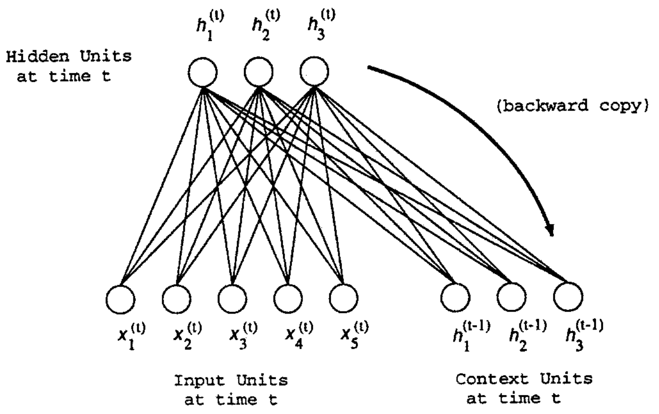
6 Learning with non-grammatical representations

In grammatical inference, formal grammars or finite automata are usually used to represent the unknown languages. There are many other forms of representations which define languages. A typical example of representations which are not in the form of grammars is *regular expressions* for regular languages. Brāzma and Čerāns [17] have studied efficient identification of regular expressions from good examples. Arikawa et al. [15] have considered *elementary formal systems*, a variant of logic programs, for identification of context sensitive languages.

In this section, we study two non-grammatical representation classes which are very hot and interesting topics in machine learning: one is simple recurrent neural networks and the other is case-based representations.

6.1 Connectionist approach

In neural network studies, recurrent neural networks have been shown to have the potential to encode *temporal* properties of a sequence of inputs. There have been proposed many recurrent neural network models [28] for dealing with temporal sequences, and here we consider variations of the *simple recurrent neural network* introduced by Elman [22]. In addition to the input and hidden units, the architecture of simple recurrent networks has an extra hidden layer of *context units* which acts as the memory or the internal state of the network (Figure 4). Thus the simple recurrent network is a two-layer feedforward network augmented by the context units and the feedback connections to context units. There has been a great deal of interest in training simple recurrent networks to recognize grammars and simulate finite automata [24].



$h_1^{(t)}$ $h_2^{(t)}$ $h_3^{(t)}$
Hidden Units at time t
(backward copy)
$x_1^{(t)}$ $x_2^{(t)}$ $x_3^{(t)}$ $x_4^{(t)}$ $x_5^{(t)}$
$h_1^{(t-1)}$ $h_2^{(t-1)}$ $h_3^{(t-1)}$
Input Units at time t
Context Units at time t

Fig. 4. Simple recurrent neural network.

Sakakibara and Golea [51] have proposed the simple recurrent network of Figure 4. The random variables represented by the input units take real values in $\mathcal{R}$, and the hidden variables represented by the hidden units take values in $\{0, 1\}$. The hidden variables represented by context units also take values in $\{0, 1\}$. The context units simply hold a *copy* of the activations (state) of the hidden units from the previous time step. Thus the next state of the hidden units is determined by the inputs and the state of the context units, the latter is equal to the previous state of the hidden units. From the finite-automata point of view, this dynamic structure is a finite-state machine and the simple recurrent network represents a state-transition function. Hence simple recurrent networks should be able to perform the same type of computations as finite automata and solve grammatical inference problems.

Sakakibara and Golea [51] have proposed these simple recurrent neural networks as probabilistic models for representing and predicting time-sequences, and shown that the model can be viewed as a generalized hidden Markov model with distributed representations. First, the state transition and the output probability functions are nonlinear. Second, the model can deal with high-dimensional, real valued vectors as output symbols. Third, it has an efficient learning algorithm using dynamic programming based on gradient descent (the algorithm can be seen as an extension of back-propagation). Moreover, compared to the previous attempts to link neural nets and HMM, the present model is more appealing because it does not require a specifically tailored architecture, e.g. *second order connections* where the multiplication operation is used between connection weights [24].

The model of Sakakibara and Golea [51] provides a new probabilistic formulation of learning in simple recurrent networks. They have presented some very preliminary simulation results to demonstrate the potential capabilities of the model. The very simple test uses a simple recurrent network with one input, two hidden units, and two context units to learn the periodic sequence "$-2, 0, 2, -2, 0, 2, -2, 0, 2, \ldots$". The learned recurrent network is shown in Figures 5 and 6. It is easy to see that the network represents a probabilistic finite automaton (HMM) and that each binary vector in the hidden layer corresponds to a state in the automaton. The output probability distribution is a time-varying (state-dependent) mixture of four basic Gaussians with variance $\sigma = 1$ and means 0, w_1, w_2, and $w_1 + w_2$. It is interesting to see how the learned recurrent network encodes the state transition function and the output function of the finite automaton in a distributed manner. For example, the learned network starts with two initial states, represented by the vectors "$(0, 0)$" and "$(0, 1)$", which have significant initial probabilities, and then repeats a sequence of state transitions: "$(0, 1) \to (1, 0) \to (1, 1) \to (0, 1)$".

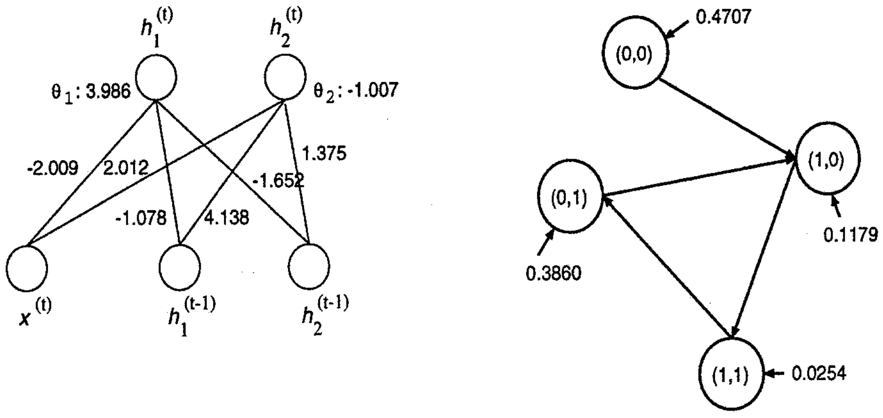


Fig. 5. The learned RNN and its equivalent probabilistic FA.

Golea et al. [27] have also presented another experimental results that use simple recurrent networks for time series prediction, and shown that the learned network is robust for outliers in noisy time sequences.

Giles et al. [24] has enhanced simple recurrent networks by connecting to an external analog stack memory. It is called a ***neural net pushdown automaton***, and manipulates the operations "push" and "pop" of the external stack and reads the top of the stack. They have tested its ability to learn some simple context-free grammars.

6.2 Case-based representation and learning

Case-based reasoning is deemed an important technology to alleviate the bottleneck of knowledge acquisition in Artificial Intelligence. In case-based reasoning,

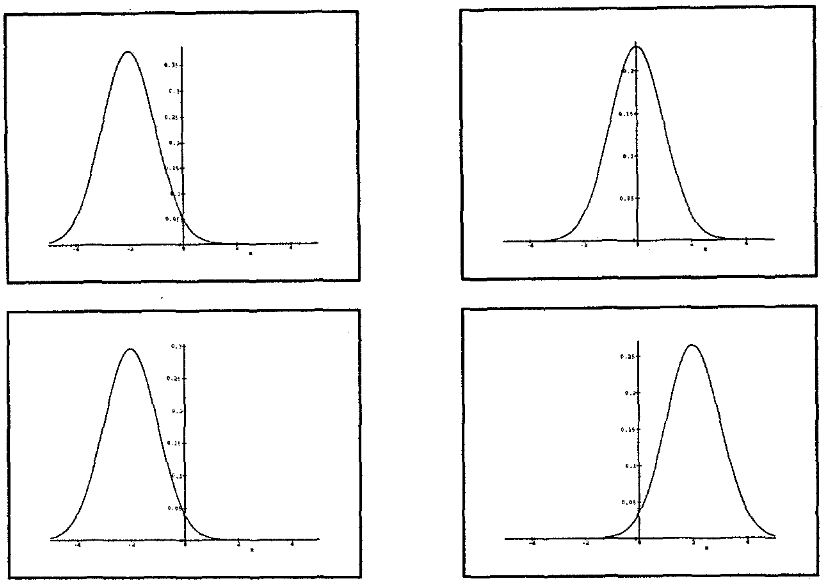


Fig. 6. Output probability distributions for state transitions: left-upper for $(0,0) \rightarrow (1,0)$, left-lower for $(0,1) \rightarrow (1,0)$, right-upper for $(1,0) \rightarrow (1,1)$, and right-lower for $(1,1) \rightarrow (0,1)$

knowledge is represented in the form of particular cases with an appropriate similarity measure rather than any form of rules. The main task of case-based learning is to collect good cases which will be stored in the case base for describing knowledge and classifying unknown examples [3]. Thus, case-based learning algorithms do not construct explicit generalizations from examples which most other supervised learning algorithms derive.

A *similarity measure* σ on Σ^* which defines a similarity between two strings is a computable function from $\Sigma^* \times \Sigma^*$ to real interval $[0,1]$. A *case base* CB is a finite subset of $\Sigma^* \times \{0,1\}$. We call a case $(w,1)$ in CB a *positive case* and $(w,0)$ a *negative case*. The language $L(CB,\sigma)$ represented by a similarity measure σ and a finite case base CB is defined as follows.

$$\begin{aligned} L(CB,\sigma) = \{w \in \Sigma^* \mid \exists (u,1) \in CB\ [\ \sigma(u,w) > 0 \wedge \\ \forall (v,0) \in CB\ [\ \sigma(u,w) > \sigma(v,w)]]\} \end{aligned}$$

We restrict all positive cases to be taken from the unknown language and all negative cases to be taken from the complement of the language.

A formal framework for case-based learning has recently been developed by Jantke and Lange [31] in an inductive inference manner. Sakakibara et al. [52] have investigated the power and the limitations of such case-based learning algorithms for formal languages in this framework. They have first shown that any indexable class of recursive languages is case-based representable, but many

Grammar

$$G = \left\{ \begin{array}{l} S \to AB,\ B \to cBd, \\ A \to aAb,\ B \to cd, \\ A \to ab \end{array} \right\}$$

Case-based representation

$$\underbrace{\left\{ \begin{array}{l} (abcd, 1) \\ (aabbccdd, 1) \\ (abcccddd, 1) \\ (aac, 0) \\ (bbb, 0) \end{array} \right\}}_{\textit{case base}} + \underbrace{\sigma(x, y)}_{\textit{similarity measure}}$$

Fig. 7. An example of grammatical representation and case-based representation for the language $\{a^m b^m c^n d^n \mid m, n > 0\}$.

classes of languages including the class of all regular languages are not case-based learnable with a fixed universal similarity measure, even if both positive and negative examples are presented.

Theorem 10 [52]. *Let $\mathcal{L}$ be any indexed class of recursive languages. There is a universal similarity measure σ such that every language L in $\mathcal{L}$ can be represented by σ and a finite case base CB of positive and negative cases, i.e., $L = L(CB, \sigma)$.*

For a complete presentation s and a natural number n, let $s_{\leq n}$ denote the initial segment of s of length n. A class of languages $\mathcal{L}$ is *case-based learnable (in the limit) from complete presentation* if and only if there are an algorithm M and a similarity measure σ such that for all $L \in \mathcal{L}$ and for all complete presentation s of L, there exists some case base CB:

1. $\forall n \in \mathbb{N} : M(s_{\leq n}) = CB_n$ is defined,
2. $\forall n \in \mathbb{N} : \emptyset \subseteq CB_1 \subseteq \{(w_1, l_1)\}$ and $CB_n \subseteq CB_{n+1} \subseteq CB_n \cup \{(w_{n+1}, l_{n+1})\}$,
3. $\lim_{n \to \infty} M(s_{\leq n}) = CB$,
4. $L = L(CB, \sigma)$.

Theorem 11 [52]. *Let $\mathcal{L}$ be the class of all finite and all co-finite languages. Then $\mathcal{L}$ is not case-based learnable from complete presentation.*

Next Sakakibara et al. [52] have considered a framework of case-based learning where the learning algorithm is allowed to learn similarity measures, too. An interesting and important method for learning similarity measures is given by adopting weighting scheme for cases like the weighted nearest neighbor algorithm. This scheme is based on the idea that some cases stored within the case base are more reliable than others. This can be accomplished with the weights in similarity measures: reliable strings are given larger weights making them more similar to strings in the target domain. Then by allowing only to learn parameters of the weights in the similarity measures, they have shown that any indexable class of recursive languages is case-based learnable. This implies, in particular, that all context-free languages are case-based learnable by collecting cases and learning parameters of the similarity measure.

7 Conclusions

We have reviewed many recent advances in the grammatical inference research. Grammatical inference is considered a main subject of inductive inference, and grammars are important representations to be investigated in machine learning from both theoretical and practical points of view. In particular, recent research activities appeal more to the practical aspects such as computational linguistics and molecular sequence processing.

Since stochastic modeling would strongly be required for practical applications, an important future problem is to find efficient algorithms which solve both problems of determining the structure of the grammar and estimating the probabilistic parameters on identifying stochastic grammars. Those algorithms should be guaranteed theoretically for their correctnesses of identifiabilities and efficiencies. Other interesting topic which we have not dealt with in this article is Genetic Search for identification of the grammars, i.e., a search using genetic algorithm techniques for finding (approximating) the target grammar. Some works (e.g., [21]) have been done to see the effectiveness of genetic search for grammatical inference problems. Finally, the formal language domain (in particular, DFAs) has been studied quite well in the PAC learning model (e.g., [39]) while we have reviewed only a few works for PAC learnabilities of formal languages.

Acknowledgements

We would like to thank Mostefa Golea, Takashi Yokomori, Takeshi Koshiba and Mitsuhiko Toda for their reading the draft and providing many valuable comments. Especially, Mostefa has helped very much for improving the paper. We also greatly thank our colleagues at the Machine Learning group at Fujitsu Labs., Hiroki Ishizaka (currently, at Kyushu Institute of Technology), Takeshi Koshiba, Masahiro Matsuoka, Yuji Takada, and Takashi Yokomori (currently, at University of Electro-Communications) for their excellent contributions to grammatical inference and enjoyable discussions.

References

1. N. Abe and H. Mamitsuka. A new method for predicting protein secondary structures based on stochastic tree grammars. In *Proceedings of 11th International Conference on Machine Learning*, 1994.
2. N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.
3. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
4. A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing.* Prentice Hall, Englewood Cliffs, N.J., 1972.
5. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

6. D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
7. D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.
8. D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.
9. D. Angluin. Learning regular sets from queries and counter-examples. *Information and Computation*, 75:87–106, 1987.
10. D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
11. D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.
12. D. Angluin. Computational learning theory: survey and selected bibliography. In *Proceedings of 24th Annual ACM Symposium on Theory of Computing*, pages 351–369. ACM Press, 1992.
13. D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of 23rd Annual ACM Symposium on Theory of Computing*, pages 444–454. ACM Press, 1991.
14. D. Angluin and C. H. Smith. Inductive inference : Theory and methods. *ACM Computing Surveys*, 15(3):237–269, 1983.
15. S. Arikawa, T. Shinohara, and A. Yamamoto. Elementary formal systems as a unifying framework for language learning. In *Proceedings of 2nd Workshop on Computational Learning Theory*, pages 312–327. Morgan Kaufmann, 1989.
16. J. K. Baker. Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.
17. A. Brāzma and K. Čerāns. Efficient learning of regular expressions from good examples. In *Proceedings of 4th International Workshop on Analogical and Inductive Inference (AII'94)*, Lecture Notes in Artificial Intelligence 872, pages 76–90. Springer-Verlag, 1994.
18. A. Burago. Learning structurally reversible context-free grammars from queries and counterexamples in polynomial time. In *Proceedings of 7th Workshop on Computational Learning Theory (COLT'93)*, pages 140–146. ACM Press, 1994.
19. R. C. Carrasco and J. Oncina, editors. *Proceedings of Second International Colloquium on Grammatical Inference (ICGI-94)*, Lecture Notes in Artificial Intelligence 862. Springer-Verlag, 1994.
20. S. Crespi-Reghizzi. An effective model for grammar inference. In B. Gilchrist, editor, *Information Processing 71*, pages 524-529. Elsevier North-Holland, 1972.
21. P. Dupon. Regular grammatical inference from positive and negative samples by genetic search: the GIG method. In *Proceedings of Second International Colloquium on Grammatical Inference (ICGI-94)*, Lecture Notes in Artificial Intelligence 862, pages 236–245. Springer-Verlag, 1994.
22. J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225, 1991.
23. A. F. Fahmy and A. W. Biermann. Synthesis of real time acceptors. *Journal of Symbolic Computation*, 15:807–842, 1993.
24. C. L. Giles, G. Z. Sun, H. H. Chen, Y. C. Lee, and D. Chen. Higher order recurrent networks & grammatical inference. In *Advances in Neural Information Processing Systems 2*, pages 380–387. Morgan Kaufmann, 1990.
25. E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

26. E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.
27. M. Golea, M. Matsuoka, and Y. Sakakibara. Unsupervised learning of time-varying probability distributions using two-layer recurrent networks. Unpublished manuscript, 1995.
28. J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.
29. L. Hunter. *Artificial Intelligence and Molecular Biology.* AAAI Press/MIT Press, 1993.
30. H. Ishizaka. Polynomial time learnability of simple deterministic languages. *Machine Learning*, 5:151–164, 1990.
31. K. P. Jantke and S. Lange. Case-based representation and learning of pattern languages. In *Proceedings of 4th Workshop on Algorithmic Learning Theory (ALT'93)*, Lecture Notes in Artificial Intelligence 744, pages 87–100. Springer-Verlag, 1993.
32. T. Koshiba. Typed pattern languages and their learnability. In *Proceedings of 2nd European Conference on Computational Learning Theory (EuroCOLT'95)*, Lecture Notes in Artificial Intelligence 904, pages 367–379. Springer-Verlag, 1995.
33. A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.
34. P. D. Laird. A survey of computational learning theory. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence - A Sourcebook*, pages 173–215. Elsevier Science Publishers, 1990.
35. K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.
36. L. S. Levy and A. K. Joshi. Skeletal structural descriptions. *Information and Control*, 39:192–211, 1978.
37. E. Mäkinen. On the structural grammatical inference problem for some classes of context-free grammars. *Information Processing Letters*, 42:193–199, 1992.
38. L. Miclet. Grammatical inference. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition - Theory and Applications*, pages 237–290. World Scientific, 1986.
39. S. Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable? In *Proceedings of 2nd Workshop on Algorithmic Learning Theory (ALT'91)*, pages 139–150. Japanese Society for Artificial Intelligence, Ohmsha, Ltd, 1991.
40. J. Oncina, P. Garcia, and E. Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, 1993.
41. F. Pereira and Y. Schabes. Inside-outside reestimation for partially bracketed corpora. In *Proceedings of 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, 1992.
42. L. Pitt. Inductive inference, DFAs, and computational complexity. In *Proceedings of AII-89 Workshop on Analogical and Inductive Inference (Lecture Notes in Computer Science, 397)*, pages 18–44. Springer-Verlag, 1989.
43. L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In *Proceedings of 21st Annual ACM Symposium on Theory of Computing.* ACM Press, 1989.
44. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77(2):257–286, 1989.

45. R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.
46. D. Ron and R. Rubinfeld. Learning fallible deterministic finite automata. *Machine Learning*, 18:149–185, 1995.
47. Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76:223–242, 1990.
48. Y. Sakakibara. On learning from queries and counterexamples in the presence of noise. *Information Processing Letters*, 37:279–284, 1991.
49. Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.
50. Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22:5112–5120, 1994.
51. Y. Sakakibara and M. Golea. Simple recurrent networks as generalized hidden markov models with distributed representations. Unpublished manuscript, 1995.
52. Y. Sakakibara, K. P. Jantke, and S. Lange. Learning languages by collecting cases and tuning parameters. In *Proceedings of 5th International Workshop on Algorithmic Learning Theory (ALT'94)*, Lecture Notes in Artificial Intelligence 872, pages 532–546. Springer-Verlag, 1994.
53. Y. Sakakibara and R. Siromoney. A noise model on learning sets of strings. In *Proceedings of 5th Workshop on Computational Learning Theory (COLT'92)*, pages 295–302. ACM Press, 1992.
54. D. B. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, Nov.-Dec. 1992.
55. T. Shinohara. Inductive inference from positive data is powerful. In *Proceedings of 3rd Workshop on Computational Learning Theory*, pages 97–110. Morgan Kaufmann, 1990.
56. A. Stolcke and S. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *Proceedings of Second International Colloquium on Grammatical Inference (ICGI-94)*, Lecture Notes in Artificial Intelligence 862, pages 106–118. Springer-Verlag, 1994.
57. Y. Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28:193–199, 1988.
58. Y. Takada. A hierarchy of language families learnable by regular language learners. In *Proceedings of Second International Colloquium on Grammatical Inference (ICGI-94)*, Lecture Notes in Artificial Intelligence 862, pages 16–24. Springer-Verlag, 1994.
59. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.
60. T. Yokomori. Polynomial-time learning of very simple grammars from positive data. In *Proceedings of 4th Workshop on Computational Learning Theory (COLT'91)*, pages 213–227. Morgan Kaufmann, 1991.
61. T. Yokomori. Learning nondeterministic finite automata from queries and counterexamples. In Furukawa, Michie, and Muggleton, editors, *Machine Intelligence 13*, pages 169–189. Oxford Univ. Press, 1994.
62. T. Yokomori. On polynomial-time learnability in the limit of strictly deterministic automata. To appear in *Machine Learning*, 1995.
63. T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. GOSLER-Report 26, TH Leipzig, FB Mathematik und Informatik, 1994.