1
Introduction

> The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.
>
> ***Albert Einstein***

> Todos os problemas são insolúveis. A essência de haver um problema é não haver uma solução. Procurar um facto significa não haver um facto. Pensar é não saber existir.
>
> ***Fernando Pessoa**, Livro do Desassossego, **107***

The problem of inducing, learning or inferring grammars has been studied now for some time, either as a purely mathematical problem, where the goal is to find some hidden function, or as a more practical problem of attempting to represent some knowledge about strings or trees through a typical representation of sets of trees and strings, such as an automaton or a grammar.

1.1 The field

Historically, one can trace back the problem to three main different sources: the first is computational linguistics, the second is inductive inference and the third is pattern recognition. The historical lineage, to be fair, requires us to discuss the important contribution of at least two other fields, not entirely independent of the three historical ones: machine learning and bio-informatics.

1.1.1 (Computational) linguistics

The key question of language acquisition (meaning the acquisition by a child of its first language), when considered as a formal language learning task, takes its root in Noam Chomsky's pioneering work. The fact that the human being should be able to discover the syntactic representations of language was key to many of the formalisms introduced to define formal languages, but also to study grammatical inference. Since that time there has been continuous research activity linked with all aspects of language learning and acquisition in the many topics related with computational linguistics.

A special idea of interest that emerged from the early work is identification in the limit, as the first paradigm for learning languages, possibly because of a strong belief that probabilities were not a plausible answer. The particular situation is that of *identifying* a *target* language from *text*, i.e. only strings from the language, also called positive examples.

There is an important thing we should insist on here, which is that one should make a distinction between *language learning* and *language acquisition*: it is generally admitted that language acquisition refers to the first language a child learns, whereas *language learning* covers the learning of another language, given a first language to support the thought and the learning process.

A second point of interest is that there is a general agreement that negative examples should not be part of the learning process even if some type of interaction between the learner and the teacher (through corrections for example) can be added.

A number of problems related with grammatical inference have been addressed in this field: parsing with formalisms that may be much more complex than grammars from formal language theory, tagging, solving many grammatical tasks... An interesting point concerns the fact that probabilistic settings have not been very popular in computational linguistics. A notable exception concerns the question of language modelling: language models intervene when different sequences of individual words or utterances have been found and one wants to decide which is the one which most probably is an actual sentence of the chosen language. The normal way to solve this is through some local decisions ('given the last two words, the new word is most likely to be...') but in that case the long-term dependencies are not taken into account. One of the major challenges to grammatical inference scientists is to come up with methods that learn these long-term dependencies.

1.1.2 Inductive inference

Inductive inference is about finding some unknown rule when given some elements of an infinite presentation of elements. Take for instance the problem of guessing the next element of the following increasing sequence: 2, 3, 5,... In this 'game' one does not actually need to find the actual function (no one is explicitly asking for that) but it is easy to see that without getting the function right we will continue to be in a situation of just guessing as the game goes on: if we don't find the actual function (or at least a function that *is like* the generating device), the task of *consistently* predicting the next element is impossible. Notice also that being right just once doesn't *prove* anything.

One of the key issues in this example is that we are not only interested in finding some mechanism – or algorithm – to do the learning, but we will also want to study the parameters of the problem: how do we get hold of our data? In this case it just arrives in some specific order, but that may not be systematically the case. How fast can we learn? How do we know that we have learnt?

Inductive inference is the research area interested in these issues. Obviously, the rules of the game may be very complex as one can play with the different variables of the problem:

- the class of functions,
- the way the data is presented,
- the rules used to measure success.

Grammatical inference is only concerned with a small sub-case of this problem: the class of functions corresponds to grammars generating or recognising strings, trees, graphs or other structured objects. On the other hand, all types of presentations may be of interest. In certain cases we will be given examples and counter-examples; in others, only some strings that belong to the language. In certain cases the learning algorithm will have to deal with examples arriving one by one, in some predefined order (or not); in another case we will be asked for a batch treatment with all the data given at once; and in an interactive setting the data will be sampled or 'bought' by the learning algorithm itself. If anything, motivation will probably reside in the fact that these presentations may correspond to practical issues. And more importantly, success will depend on some provable convergence with only a limited quantity of resources, whether computation time, quantity of examples or number of accesses to an oracle.

The inductive inference field has continued to introduce and analyse key ideas and has led to the development of a field known as algorithmic learning theory. The name of the field is possibly slightly misleading, because in most cases the algorithms are not developed with an interest in breaking polynomial boundaries but more with the mind set on showing what is decidable and what is not.

1.1.3 Pattern analysis, pattern recognition

In the problem of recognising handwritten characters (for example the postal codes on an envelope), we may be given images of each of the possible letters and digits, construct some sort of a model for each symbol, and then, when a new character is presented, we can compare the new character to the models and find the model that fits best. This idea requires robust models, a training phase to set the parameters of the models and some way to measure how adequate a model is for a given character.

One may also consider that a character, once pixellated, can be described by a string (if we use either four or eight directions). To take into account the variety of lengths and the variations of figures, one possibility is to associate with the character 'A' a language of all strings corresponding to this character. Then by parsing a new string, one should be able to decide if the current written character is an 'A' or a 'B'. There are obviously many variations of this question, where, for example, the language may include probabilities so that eventual intersections can be dealt with. There will also be a need to study distances between strings or between strings and languages to take better decisions.

The above is one of the many attempts to use grammar induction in pattern recognition. We use here the term *grammar induction* and not *grammatical inference*: there is actually

no reason to believe that there is a hidden grammar to be discovered. In this case we are only believing that somehow a grammar can represent in a reasonable way a set of strings corresponding to the letter 'A'.

A second point one can make is that if the task we describe is a pattern recognition task, since the induced grammar has a meaning *per se*, it is tempting to analyse this grammar, and therefore to consider that grammatical inference techniques enable us to discover *intelligible* patterns, and thus that they permit us to tackle *pattern analysis* tasks.

1.1.4 Machine learning

From the inductive inference perspective, classes and paradigms were analysed with a distinct mathematical flavour.

On the other hand, more pragmatically oriented researchers were working on practical problems where techniques were tested enabling them to 'induce' an automaton or grammar from some data.

The meeting point was going to be provided by the work done in another field, that of machine learning.

Machine learning has developed as an important branch of artificial intelligence. The great idea of machine learning is to enable a system to get better through some form of interaction with its environment in which new knowledge is extracted from experience and integrated in the system. Some examples are learning from observation, from examples, from mistakes, and from interaction with a teacher.

Work consisted of developing ideas, algorithms and even theories of what was learnable and what was not, whilst at same time considering important applications where *very nice heuristics* were not going to provide a suitable answer: provable algorithms were needed, in which the quantity of necessary information requested in order to be able to learn some satisfactory function had to be minimal. All the notions associated with the key words in the previous sentence need to be defined. This led to the introduction of models of learning of which the probably approximately correct (PAC) model is the best known, and unarguably the most inspiring.

These efforts to re-understand the learning algorithms through the eyes of complexity theory (polynomial time and polynomial space required), statistics (given some known or unknown distribution, can we bound the error the classifying function will make over unseen data, or can we bound the error over the possibility that our error is very small?) have also influenced the special problem of learning grammars and automata.

At first, the influence of researchers in the growing field of computational learning theory (COLT) led to some essential results and analysis around the central problem of learning deterministic finite state automata (DFA) in the important and specific setting of PAC learning. The task was proved to be impossible using results on learning from equivalence queries. The hardness of the minimal consistency problem (for DFA) was proved in 1978; but the mathematical links between complexity theory and computational learning theory led to elaboration on that result: the closely related problem of finding a small consistent

DFA (not more than polynomially larger than the smallest) was proved to be intractable. This on one hand forbade us to have hopes in Occam approaches but also opened the path to finding other negative results. These somehow seemed to *kill the field* for some time by showing mathematically that nothing was feasible even in the case of what most formal language theoreticians will consider as the easiest of cases, that of DFA.

The introduction of alternative learning models (like active learning), the capacity of probabilistic automata and context-free grammars to model complex situations and the necessity to learn functions with structured outputs allowed a renewal of interest in grammatical inference techniques and are some factors indicating that future cross-fertilisation between grammatical inference and machine learning scientists could be a key to success.

1.1.5 Computational biology

The basic elements in computational biology are strings or sequences describing DNA or proteins. From these a number of problems require the analysis of sets of strings and the extraction of rules. Among the problems that have suggested that grammatical inference could be useful, we can mention the one of secondary structure prediction. Context-free grammars can describe partly some well-known structures, which has motivated several attempts in the direction of learning context-free grammars. But, because of the size of the data, other researchers argued that grammars were going to prove to be too complex for computational biology. Pattern languages have been favoured by these researchers, and works in the direction of extracting patterns (or learning pattern languages) have been proposed.

There are a number of good reasons for wanting to use grammatical inference in this setting, but also a number of reasons why the results have so far been on the whole disappointing.

It is clear that we are dealing with strings, trees or graphs. When dealing with strings, we will typically be manipulating a 4-letter alphabet $\{A, T, G, C\}$ (standing for the four basic nucleotides) or a 20-letter alphabet if we intend to assemble the nucleotides 3 by 3 into amino-acids in order to define proteins. There are trees involved in the patterns or in the secondary structure, and even graphs in the tertiary structure.

Then, on the other hand, the strings are very long, and the combination 'very long strings' + 'recursivity' + 'no noise' makes the task of learning quite hard. One particularity of formal languages is that, if they do allow recursive definitions, they are not very robust to noise, especially when the noise is defined through edit operations. There are nevertheless a number of attempts to learn automata, grammars and transducers for biological applications.

1.1.6 Grammatical inference as an independent field

In 1994 the first *International Colloquium in Grammatical Inference* took place, arranged as a follow-up to a workshop that had been organised the year before. From then on,

grammatical inference has emerged as an independent field. Little by little the conference acquired its rules and the community organised itself. This did not lead to the researchers working in the field isolating themselves: terms like *grammatical inference* and *grammar induction* have progressively been accepted in many communities and grammatical inference is today a meeting point between researchers from many areas, with very different backgrounds, most of whom have a primary research interest in one of many other fields. Nevertheless, whether for children's language acquisition, automatic music composition or secondary structure prediction (to name just three very different tasks), the objects remain common and the algorithms can be similar.

1.2 An introductory example

Let us introduce the problems and techniques of grammatical inference through a simple problem from a multi-agent world, where the goal is to discover the strategy of an adversary agent.

1.2.1 The problem

Let us consider the following situation: in an interactive world an agent has to negotiate with (or against) other agents. He or she may be wanting to acquire or sell resources. The agent's decisions (to buy or not to buy) will depend on what he or she expects to win but also on the other agent's attitude: should he or she keep on negotiating in order to reach a better price? Is the opponent going to accept the lower price?

Let us suppose that the adversary agent is driven by a finite state machine which describes its *rational strategy* (see Figure 1.1 for a trivial first example). The automaton functions as follows: at any one time the agent is in a given state and will act according to the label of that state. At the same time the agent discovers the action his or her opponent makes and moves on to the state reached through reading the action as a label of an

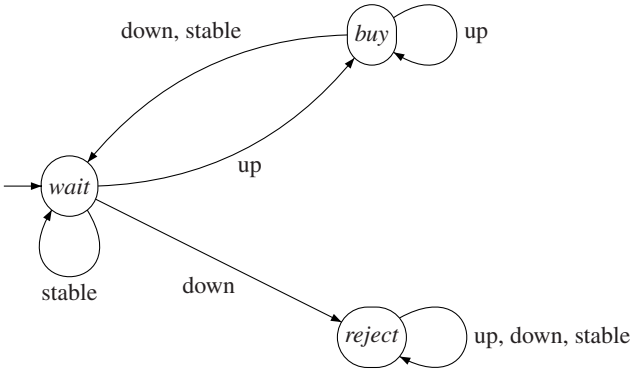


Fig. 1.1. A primitive rational strategy for a buying situation.

Table 1.1. *The game matrix for the buying game.*

	buy	*wait*	*reject*
up	5	1	-80
stable	5	-1	-80
down	10	3	-80

edge. The agent is then ready to proceed. Obviously the model is rather primitive and some sort of non-determinism/probabilistic outcome would be better suited. If we consider the automaton depicted in Figure 1.1 this agent will start by being doubtful (action *wait*). If our first offer is a raise (action up) then the agent will start buying (action *buy*), but we will need to raise each turn (action up) for him or her to stick to the buying policy, which is presumably what we want. If not, the agent will get back into the neutral initial state from which, if we lower our offer, he or she will actually get into a refusal state (action *reject*) in which no more negotiation can take place.

1.2.2 The gain matrix and the rules of the game

But there is more to it than just trying to see what the opponent is doing. Finding the right action for an optimised immediate gain is one thing; ensuring long-term gain is another! This is the question we are interested in now. Notice also that the reasons for which the adversary adopts a particular policy and his own gain are not of direct importance here, since only our own long-term gain matters.

So the sort of game we are considering is one where each player has to choose between a finite number of *moves* or *actions*. Both players play their move simultaneously. To simplify, if we have players A and B, each player is allowed to play moves from Σ_A and Σ_B respectively, where Σ denotes the alphabet, or set of allowed moves.

Each time an event takes place (an event being a simultaneous action by each agent) there may be a gain or a loss. This is usually represented through a gain table, such as in Table 1.1 where $\Sigma_A = \{\text{up, down, stable}\}$ and $\Sigma_B = \{\textit{buy}, \textit{wait}, \textit{reject}\}$: we have represented what we expect to gain out of an event. For example if simultaneously we (as player A) raise and our opponent (player B) is *neutral* (*wait*), then our gain is of +1; in all cases if the game is terminated by the opponent (*reject* state) our loss will be of -80.

1.2.3 The prisoner's dilemma

The best known and studied variant of the above game is called the prisoner's dilemma. The story can be told as follows:

Table 1.2. *The game matrix.*

	s	*a*
s	1	5
a	3	0

There has been a crime for which two persons have been arrested. In the morning, the judge addresses each of the suspects separately: *"We believe you are guilty but have no proof of this. So, we propose the following deal:*

If you admit guilt but your colleague does not, then you will be considered as an informer, get out free and your colleague will be jailed for five years.

If you decide to remain silent, on one hand your colleague might decide to be less stupid and accept the offer (remember that in this case you will suffer the entire burden alone, and go to jail for five years!).

If you both plead guilty we won't need an informer, but you both get a three-year sentence.

In the very unlikely case where you should both choose to remain silent, we have against you the fact that you were found doing grammatical inference in the dark, so you will both suffer a one-year sentence. . . "

This is a symmetrical game as both the opponents have the same gain matrix represented in Table 1.2: we will denote being **s**ilent by '*s*', whereas '*a*' stands for **a**dmitting and thus defecting. By analysing the game it will appear that the Nash equilibrium[†] is reached when both players decide to defect (admit), reaching a far from optimal situation in which they both get imprisoned for three years! The game has been studied extensively in game theory; a simplified analysis goes as follows:

- Suppose the other person remains silent. Then if I am also silent I get one year in prison, whereas if I admit I get out free: better to admit.
- Suppose on the contrary he admits. If I am silent I will spend the next five years in prison: better admitting since in that case it will only be three.

The natural conclusion is therefore to admit, in which case both players reach the *bad* situation where they both 'pay' three years; if they had cooperated they would get away with a one-year sentence. The game has been much more thoroughly analysed elsewhere. We may now ask ourselves what happens when the game is iterated: the goal is only to score points and to score as many[‡] as possible over time. The game is played over and over against the same opponent.

Suppose now that the opponent is a machine. More specifically interesting to us is the case where this machine follows a *rational strategy*. In this case the object is no longer to win an individual game but to win in the long term. This is defined by means of a gain

[†] A Nash equilibrium is met when no player can better his situation for the opponent's current move.
[‡] In this case 'as many' requires spending as few years as possible in prison!

function computing the *limit of means*: the best strategy, against a fixed adversary, will be the one which has the highest mean gain, for an arbitrarily long game. With this, a finite number of initial moves which may seem costly will, in the long run, not count. Formally, the gain of strategy one (S_1) playing against strategy two (S_2) is defined by first noting that if both strategies are deterministic, a unique game is described. This unique game is an infinite string over an alphabet $\Sigma_A \times \Sigma_B$: $\langle m_0^A, m_0^B \rangle, \ldots \langle m_i^A, m_i^B \rangle, \ldots$ where $\langle m_i^A, m_i^B \rangle$ is the combination of the ith moves made by players A and B. In such a sequence, let us denote by $G[m_i^A, m_i^B]$ the gain for player A at step i. This in turn leads to a unique outcome described by the game matrix: $g(S_1, S_2, i) = \sum_{j \leq i} G[m_i^A, m_i^B]$.

$$g(S_1, S_2) = \lim_{n \to \infty} \frac{\sum_{j \leq n} g(S_1, S_2, n)}{n}$$

Remember that in this setting we are not directly interested in how much our opponent may make out of the game, only in what we are going to make.

1.2.4 What is a rational strategy?

We will say that a strategy is rational if it is dictated by a Moore machine. Informally (formal definitions will be given in Chapter 4) a Moore machine is a finite state machine whose edges are labelled by the actions made by an opponent and whose states have an output corresponding to our action when in the given situation. Consider for instance the rational strategy depicted in Figure 1.2. This corresponds to the iterated prisoner's dilemma as described above. Then it follows that an agent using this strategy will start by playing a, independently of anything else (remember that both players play simultaneously, so this is quite normal).

Now the agent's next call will depend on our first move. If we are silent, then the agent follows the transition labelled by s into the middle state: his or her move is deterministically a result of our move. His or her second action will be to be silent also, since that is what is indicated in that state. If on the contrary we had defected (action a), then the agent would have remained in the left-hand state and would repeat a.

Again, from Figure 1.2 it follows that if the first three calls in some game are $\langle \mathtt{a}, a \rangle, \langle \mathtt{a}, a \rangle, \langle \mathtt{s}, a \rangle$ then the agent using this strategy will necessarily play s as his or her fourth move since having parsed his or her adversary's first three moves (the string aas) the agent is in the central state corresponding to output s.

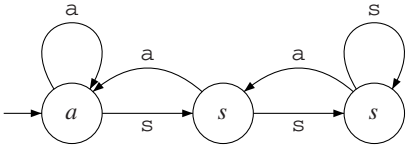
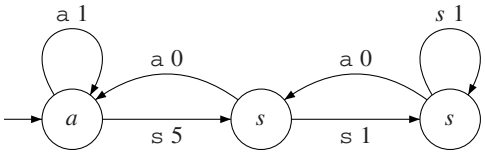


Fig. 1.2. A rational strategy.

	s	a
s	1	5
a	0	3

(a) Gain matrix



(b) The adversary graph, labelled.

Fig. 1.3. Using the graph to find the best path.

1.2.5 Beating a rational strategy that we know

We suppose that the strategy of the opponent is rational, i.e. it is given by a deterministic finite automaton as above (Figure 1.2). The first question is then: can we imagine an optimal strategy against it?

Suppose first that we know the opponent's strategy: then we can use game theory to compute the winning strategy. Let us consider for that the opponent's graph in which we value the edges by our own gain. Take for example the situation from Figure 1.3 and the strategy represented in Figure 1.2. The edges are weighted according to our values from the game matrix recalled in Figure 1.3(a). The result is shown in Figure 1.3(b).

What is a winning strategy in this case? As the adversary has imposed moves, it consists of finding a path that has minimal average weight. As the graph is finite this path is obtained by finding a cycle of minimum mean weight and finding the path that is best to reach this cycle. Both problems can be solved algorithmically.

Algorithm 1.1 computes the best strategy (in the sense of our criterion) in polynomial time. In this case the best strategy consists of being silent even when knowing that the opponent is going to defect, then being silent once more, and then alternating defection and silence in order to beat the more natural collaborative approach.

Algorithm 1.1: Best strategy against a rational opponent.

Data: a graph
Result: a strategy
Find the cycle of minimum mean weight;
Find the path of minimum mean weight leading to the cycle;
Follow the path and stay in the cycle

Summarising, if we know the adversary's strategy, we can run a simple algorithm that returns our own best strategy against that opponent. All that is now needed is to find the opponent's strategy, at least when it is described by an automaton!

So the question we now want to answer becomes: 'Having seen a game by this opponent can we reconstruct his strategy?'