Machine Learning, 10, 195–199 (1993)

Introduction

Case-based reasoning means reasoning based on remembering previous experiences. A reasoner using old experiences (cases) might use those cases to suggest solutions to problems, to point out potential problems with a solution being computed, to interpret a new situation and make predictions about what might happen, or to create arguments justifying some conclusion. A case-based reasoner solves new problems by remembering old situations and adapting their solutions. It interprets new situations by remembering old similar situations and comparing and contrasting the new one to old ones to see where it fits best.

Case-based reasoning is a method that combines reasoning with learning. It spans the whole reasoning cycle. A situation is experienced. Old situations are used to understand it. Old situations are used to solve its problem (if there is one to be solved). Then the new situation is inserted into memory alongside the cases it used for reasoning to be used another time.

Key to this reasoning method, then, is remembering. Remembering has two parts: integrating cases or experiences into memory when they happen and recalling them in appropriate situations later on. The case-based reasoning community calls this related set of issues the **indexing problem**. In broad terms, it means finding in memory the experience closest to a new situation. In narrower terms, we often think of it as a two-part problem:

- assigning indexes or labels to experiences when we put them into memory that describe the situations to which they are applicable, so that they can be recalled later, and
- at recall time, elaborating the new situation in enough detail so that the indexes it would have if it were in the memory are identified.

Retrieval processes, themselves, as implemented in our programs, use the new situation, any elaborations of it that have been done, and the current reasoning goal of the reasoner to find new cases. The new situation and its elaborations act as a retrieval key in finding similar cases; the reasoner's goal ensures that, of the similar cases, those that can best fulfill the need of the reasoner are recalled. Our systems anticipate the usefulness of cases when they add them to memory and label them according to that interpretation, and they interpret new situations to figure out what would be useful to recall and attempt to recall those things.

This discussion points out two capabilities that determine how well a case-based reasoner will perform. The extent to which a reasoner can anticipate the usefulness of the cases it stores in its memory is an important determiner of the capability of the case-based reasoner. Another important determiner of capability is the extent to which a reasoner can interpret a new situation and determine which kinds of cases are most likely to be useful (this process is called *situation assessment*).

Another factor that determines the capabilities of a case-based reasoner is the range of experiences it has had. In general, the broader the range of experiences a reasoner has had, the broader the range of new situations it will be able to reason about. And the more subtle the differences between the cases it has experienced, the more competent it will be at recognizing these subtle differences and making more relevant and specific inferences.

Learning, in a case-based reasoning system, is largely an emergent behavior that arises from the case-based reasoner's normal functioning; largely it is a process of accumulating new cases, but it is dependent on other processes. The extent of learning that a case-based reasoner can do is dependent on the range of experiences it has had, its ability to explain and repair failures, its ability to explain its successes, and, based on all of this, its ability to generate good indexes.

Suppose, for example, that a system is asked to solve a problem outside of its range of experience. It might be able to apply adaptation heuristics to solve the problem based on some case or cases it already knows, or it might fail. If it is able to solve the problem, entering the new situation and its solution into memory as a case will allow the reasoner to solve such problems more efficiently in the future—there will be less adaptation needed. If it fails to solve the problem, the reasoner might or might not be able to learn by itself. Whether it can learn and how much it can learn will depend on the feedback it has available to it, the explanatory procedures it has available for interpreting that feedback, and its ability to do repair.

A system that can explain its failures and figure out how it could have avoided the problem can insert the new situation into its case library, indexing it to enable recognition in the future that a failure situation is imminent and to avoid it. If, in addition, the reasoner can figure out what would fix the problem, that information can be inserted into the case, and the case can be indexed so that its repair can be recalled as a potential solution in similar situations.

Such learning has an added benefit. When failure cases are inserted into a case library, they help a reasoner realize what it should pay attention to in a situation. If a reasoner has been introduced to a variety of situations that differ in subtle ways, has been allowed to fail, and has been able to explain its failures, it has the opportunity to become quite accurate in its reasoning—failure cases it is reminded of will point out to it what it should be paying attention to as it reasons, aiming it toward an appropriate solution.

The ability to explain why solutions succeed is also advantageous for a reasoner. Explaining why solutions succeed results in indexes being chosen for a case that reflect the full range of situations in which it can be useful. Indexes chosen based on such explanations allow a reasoner to recognize opportunities for reusing its old solutions. These indexes can also given reasoners the ability to recognize opportunities for achieving background goals.

Another learning issue that some case-based reasoners address is the re-indexing of cases that are already in memory. When a case is recalled and proves inappropriate (either because it cannot be used or because it is used and results in a failure), one might want to re-index the case so that it is no longer recalled in situations such as the current one. When it is pointed out to a reasoner that it should have used a case it did not recall, that case should be re-indexed so that it will be recalled in the future in similar situations. When use of a case points out something new about it that was not known previously, it should be re-indexed to reflect that new knowledge.

In addition to the learning that happens as a natural consequence of reasoning, one can think about enhancing a case-based system with learning capabilities designed to make its components work more efficiently. For example, one could put controls on which cases are added to a case library so that retrieval can be kept as efficient as possible. One could have a reasoner learn which solutions are costly to carry out and which are inexpensive (to enhance its selection of cases), which elaborations are costly to carry out and which are inexpensive (to make situation assessment more efficient), and so on.

Consider, first, the controls one might put on accumulation of cases. Some cases added to a case library might add to the library without adding to the capabilities of a system. Such cases might or might not effect the efficiency of retrieval, depending on the retrieval algorithm. They certainly require additional storage capacity. Sometimes it is worthwhile making decisions about whether a new case belongs in the case library or not. The new case belongs if it allows the reasoner to do something it could not do before or to become more efficient at something it did before. Or, a reasoner could be proactive about accumulating cases. If a reasoner knows there are some things it is incapable of or has trouble with, it can seek to put itself in situations in which it will acquire appropriate cases, or it can look out for cases that can help it with its troublesome tasks.

Another set of enhancements that several researchers in the case-based reasoning community have addressed are those that enhance situation assessment. In situation assessment, the reasoner elaborates a situation description to make its description fit other case descriptions in its case library, to allow further traversal of memory structures, or to differentiate between several cases competing for retrieval. Some elaborations are costly, requiring either complex inference or complex action; others are less expensive. A system that keeps track of the cost of each of its elaborations and the degree to which each is advantageous can guide its situation assessment procedures in elaborating the most useful features and preferring those that are less costly.

Cases can also be used to increase general knowledge. Some case-based programs accumulate generalizations as they add to their case libraries. Such generalized knowledge is useful in controlling indexing and in making elaborations during situation assessment. Other programs keep track of the knowledge they need as they reason. They accumulate not only cases, but also general knowledge they know they need that is embedded in those cases. While these systems rely primarily on cases to reason, they each also use some other knowledge, and each has a complex control structure. When they are aware of some piece of knowledge they need to enhance their control structures or to enhance other means of reasoning, they derive that from cases also.

Overall, accumulating the right cases, extracting the appropriate knowledge from them, and making sure cases are indexed advantageously results in several different enhanced performance behaviors:

- the ability to perform in more situations (i.e., increased proficiency)
- increased efficiency in familiar situations
- increased ability to cope in problematic situations
- increased ability to take advantage of opportunities as they arise

The set of articles in this volume together address the whole range of learning issues described above. Each of the reasoners reported on in this volume accumulates cases, and as a result becomes a more efficient or proficient reasoner. PRODIGY, discussed by Carbonell and Veloso, combines case-based reasoning and general-purpose problem solving in order to plan. As PRODIGY reuses its old experiences, it adapts them to make them fit new situations and reinserts them into its case library, making both its general problem solving and its case-based reasoning more efficient over time. It fine-tunes its indexes as a result of using its cases, making its performance more proficient over time in addition.

Hammond, Converse, and Marks also report on case-based planning, but focus on a different aspect of the learning. Their systems learn to take advantage of execution-time opportunities. While is is natural to think about a case-based system explaining its failures and becoming more proficient as a result, their systems do more. They notice when they have been able to accomplish more than they were expecting to and when they were able to accomplish their goals well, and they work to create opportunities in the future for carrying out their goals equally well.

Owens' program, ANON, learns something that is of benefit to any learner working within any paradigm—what to pay attention to in the environment. A reasoner is often in a situation of needing to choose between many interpretations of a situation, and therefore, several ways to go about addressing the situation's problem. Reasoners in this state need to collect more information to distinguish between the several interpretations. But some information is more expensive to collect than others, and some is more useful than others. ANON provides a way of deciding what additional information should be collected in any particular situation and translates that into notations that allow the reasoner to weigh the utility of collecting any particular kind of information in the future.

Ram's program, AQUA, also works in situations where everything is not well known or well specified, and it uses the cases it encounters to build up knowledge (in both the forms of cases and generalizations) about a domain. In response to its experiences using its knowledge for understanding, it revises the knowledge it already has, re-indexes its cases and general knowledge, and sets itself up with goals to acquire additional knowledge that it has identified as potentially useful. Keeping track of its knowledge goals allows it to decide which, of all the cases it has encountered, are worth recording in its memory.

Finally, Krovvidy and Wee's wastewater treatment system uses case-based reasoning to enhance heuristic search when applied to a reasonably complex problem. In addition to building a system that learns to do heuristic search better, they have been able to identify necessary properties of heuristic search problems that make them amenable to the application of case-based reasoning.

This set of articles is representative of the learning research going on in the case-based reasoning community, but it does not provide full coverage. There are a number of important contributions to case-based learning left out of this set of articles. First is the set of early programs that learned through accumulation and through identifying and repairing their reasoning errors (e.g., MEDIATOR, PERSUADER, CHEF). Next are programs and projects that have been finished for several years and reported on elsewhere (e.g., PROTOS, IVY). PROTOS learns a new domain by accumulating cases and interacting with an expert when it is unsure of its interpretations. IVY determines which new cases it needs to acquire by keeping track of the problems it runs into in reasoning and setting itself up with explicit

goals to collect cases that help it do a better job of those tasks. There are also other programs and approaches that were too new to make it into this volume (e.g., CELIA), and others that address primarily indexing issues (e.g., ABBY) or adaptation processes (e.g., JULIA) that support learning but are not about learning themselves. CELIA learns by actively understanding the problem-solving behavior and explanations of an expert and by making the expert's experiences solving problems into well-interpreted cases of its own. In addition, there are a number of recent projects, too numerous to mention, that apply case-based reasoning to heuristic search problems and learn better how to search a space, avoiding paths that are prone to failure and that learn better similarity metrics for comparing cases to each other as they perform.

Janet L. Kolodner
Georgia Institute of Technology