Relational Local Iterative Compression

Laurent Orseau
UMR AgroParisTech / INRA 518
AgroParisTech
16 rue Claude Bernard, Paris, France

Compression in the program space is of high importance in Artificial General Intelligence [Sol64, Hut07]. Since maximal data compression in the general sense is not possible to achieve [Sol64], it is necessary to use approximate algorithms, like AIXI$_{t,l}$ [Hut07].

This paper introduces a system that is able to compress data locally and iteratively, in a relational description language. The system thus belongs to the anytime algorithm family: the more time spent, the better it performs. The locality property is also well-suited for AGI agents to allow them to focus on "interesting" parts of the data.

The system presented here is to be opposed to blind generate and test approaches (e.g., [Sch04, Lev73]). On the contrary to the latter, it uses information gathered about the input data to guide compression. It can be described as a forward chaining[1] expert system on relational descriptions of input data, while looking for the most compressed representation of the data.

It is composed of a description/programming language, to describe facts (and a set of weights associated with each primitive of the language), local search operators, to infer new facts, and an algorithm to search for compressed global description.

The relation operators and the search operators are domain-specific. Examples in the letter-string domain are given in the *Experiments* section. Due to lack of space, only a overview of the whole system can be given.

Description Language

The main point of this paper is to deal with *local compression*. This means that the system should be able to focus on any part of the input data, without affecting the rest of the data.

A relational language for data representation/programming is well suited for this purpose, exactly because everything (including spatial and dynamical dependencies) can be described in terms of local relations between data parts.

The language has values (numbers, characters, operators names, ...) and relations between objects (instantiations of operators on objects). What kinds of objects and operators are used depends on the domain. For an AGI, it depends on the sensors it uses, but the set of operators should form a Turing-complete language.

The *initial description* of the *world* (the input data) is the initial facts of the expert system.

[1]There can be no backward chaining, because no goal description is given.

Search Operators

The inference rules of the expert systems are called the *search operators*. A search operator takes inputs, tests them against a precondition, and when the test is passed produces outputs that are added to the fact database. The set of search operators is domain-dependent.

The exact inputs/outputs mapping is also memorized to construct a graph for the compression part of the algorithm.

The constraint imposed on search operators is that they must not lose information, i.e. that knowledge of the outputs is sufficient to reconstruct the inputs.

Algorithm

The algorithm runs like this:

1. The input data is given to the system in a simple uncompressed relational representation.
2. Each local search operator is tested in turn to create new local descriptions when possible.
3. Local descriptions are composed to create global descriptions.
4. The description (space) costs of the global descriptions are computed.
5. The less costly global description is retained.
6. Stop when a given criterion (on time, space, error, ...) is satisfied, or go back to step 2.

Finding the best current description is based on the Minimum Description Length principle [GMP05], where the cost of a description is simply the sum of the costs of each relation used. The cost of a relation is domain specific, and defined by the user.

Lossless Compression Sketch Proof

Search operators consume inputs and provide outputs (new local descriptions). If each such operator has the property that it does not lose information, i.e. that its inputs can be rebuilt from the outputs, then the global algorithm ensures that no information is lost for global descriptions. The only difficulty resides in cyclic dependencies between local descriptions, e.g. when a local description A depends on the local description B and vice-versa. To avoid such cycles, a dependency directed graph of input-output mappings created by the search operators is constructed, and any cycle is broken. The final description is composed of the relations that are on the terminal nodes of the graph. So some inputs that should have been consumed can appear in the final description because they are needed to avoid a cycle.

Experiments

The system has been tested in the letter-string domain on a set of small strings that show that compression is indeed iterative.

In the letter-string domain, the world is initially described using the following relations:
obj: binds values that describe one same "object", character, of the world (the string),
val: character value a, b, c ...
pos: position of the object in the string,
Once compression has begun, some of the following relations may be used:
neighbor: two objects are neighbors,
succ, **pred**: two values follow one another in lexicographical order
eq: two values are identical,
plus relations to describe sequences (with a initial value, a length and a succession relation) and sequences of sequences.

The letter-string domain search operators have similar names to the description relations. For example, the **eq** search operator searches the fact database for identical values. When it finds one, it creates a new fact using the **eq** relation binding the two values and adds it to the database. The **seqV** search operator searches for two objects that are neighbors and have a relation on their values and creates a new sequence of two objects, whereas **seqG** tries to merge two existing neighbor sequences that have identical succession operators.

For example, the initial string `abcxxxxdefyyyy` has a cost of 28 (given a fixed set of costs for the letter-string domain). After one pass on the loop of the algorithm, the system compresses it to a cost of 24.9, finding local relations like **neighbor** and **eq**. On the next loop step, it finds other relations like small sequences but they do not build a less costly description. On the next steps, the sequences grow, lowering the best description cost to 18.8, then 17.6 and finally 14.3, where the string has been "understood" as `(abc)(xxxx)(def)(yyyy)` with **succ** relations between interleaving sequences and **neighbor** relations between adjacent sequences.

The system also compresses non-obvious strings like `abccdedefg`, on which it lowers the initial cost of 20 to 8.3 with 7 intermediate values, finally finding the compressed representation of the sequence of sequences `((a)(bc)(cde)(defg))`.

Limitations, Perspectives and Conclusion

For the experiments in the letter-string domain, a few seconds are sufficient to find a much compressed description, but lengthening the initial strings leads to a huge combinatorial explosion. To limit the impact of such explosion, the first solution is to add ad-hoc domain-specific search operators that focus on specific "interesting" patterns in the database and are given high priority. It is also possible to add a learning strategy, for example inspired by Explanation Based Learning [DM86], since compressing is equivalent to proving. Learning would lead, with an AGI approach, to Incremental Learning (e.g. [Sch04]), using acquired knowledge to solve related problems faster. Learning could then also be used to incrementally tune the initial costs of the relation operators like **eq**.

The language used for the experiments can represent complex non-linear worlds, but the language should be augmented to Turing-completeness since for an AGI this seems to be unavoidable.

Relational local iterative compression is a novel approach to compression in the program space and could be used for many different tasks, e.g. visual scene (2D, 3D) compression/comprehension, amongst others. It may be mostly beneficial when prior domain knowledge can be used or acquired.

References

[DM86] G. Dejong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176, 1986.

[GMP05] P. D. Grünwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[Hut07] M. Hutter. Universal algorithmic intelligence: A mathematical top-down approach. In *Artificial General Intelligence*, pages 227–290. Springer Berlin Heidelberg, 2007.

[Lev73] L. A Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[Sch04] J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

[Sol64] R. J Solomonoff. A formal theory of inductive inference. *Information and Control*, 7(1):1–22, 1964.