The Cruncher: Automatic Concept Formation Using Minimum Description Length

Marc Pickett and Tim Oates

University of Maryland, Baltimore County, USA
marc@coral.cs.umbc.edu

Abstract. We present The Cruncher, a simple representation framework and algorithm based on minimum description length for automatically forming an ontology of concepts from attribute-value data sets. Although unsupervised, when The Cruncher is applied to an animal data set, it produces a nearly zoologically accurate categorization. We demonstrate The Cruncher's utility for finding useful macro-actions in Reinforcement Learning, and for learning models from uninterpreted sensor data. We discuss advantages The Cruncher has over concept lattices and hierarchical clustering.

1 Introduction

Concept formation is a form of abstraction that allows for knowledge transfer, generalization, and compact representation. Concepts are useful for the creation of a generally intelligent autonomous agent. If an autonomous agent is experiencing a changing world, then nearly every experience it has will be unique in that it will have at least slight differences from other experiences. Concepts allow an agent to generalize experiences and other data. In some applications, the concepts that an agent uses are explicitly provided by a human programmer. A problem with this is that the agent encounters problems when it faces situations that the programmer had not anticipated. For this reason, it would be useful for the agent to automatically form concepts in an unsupervised setting. The agent should be able to depend as little as possible on representations tailored by humans, and therefore it should develop its own representations from raw uninterpreted data.

One purpose of concept formation (and abstraction in general) is to concisely characterize a set of data (Wolff[9]). With this view, one can use *minimum description length* (MDL) as a guiding principle for concept formation. We have developed an algorithm, called The Cruncher, which uses this principle to form an ontology of concepts from a collection of attribute sets. The Cruncher is general in the sense that no further knowledge of the attribute sets needs to be provided.

In the past, several other methods have been proposed for concept formation. Perhaps the most common form of unsupervised concept formation is clustering. For an overview of some of these algorithms, see Fasulo[4]. A drawback to much

J.-D. Zucker and L. Saitta (Eds.): SARA 2005, LNAI 3607, pp. 282–289, 2005.
© Springer-Verlag Berlin Heidelberg 2005

of this work is that each item belongs in only one cluster. The Cruncher uses multiple inheritance, which allows it to overcome obstacles faced by strictly hierarchical classification systems such as hierarchical clustering and decision trees. For example, in Figure 1, The Cruncher describes the Penguin (which has attributes in common with Birds and Fish) as both a "bird" and an "aquatic" creature. Hierarchical clustering would force the Penguin to be in only one of these classes.

There has also been work on Ontology Formation (for example, see Gruber[6]), but much of this work is aimed at knowledge engineering, where a human's help is required to assist the ontology formation. The Cruncher is completely unsupervised in contrast. The Cruncher also allows for exceptions, which further sets it apart from most of the work in this community. The Cruncher's exceptions allows for better compression, and, for the UCI Zoo Database, allows for classifications that correspond to the human-developed classification. For example, the Platypus is described as an egg laying mammal, even though mammals are defined as not laying eggs.

The field of Formal Concept Analysis provides methods for producing Concept Lattices, which form an ontology with multiple inheritance. The main aspects that set The Cruncher apart from this work are: first, due to the rigidity of Formal Concept Analysis, exceptions are not allowed as they are for The Cruncher, and second, MDL is not a driving factor in producing the concept lattices. As we demonstrate in Section 3, The Cruncher provides better compression than standard Concept Lattice layout algorithms, such as that described in Cole[3]. For an overview of the field of Formal Concept Analysis see Ganter and Wille[5].

The idea for The Cruncher grew out of "PolicyBlocks", an algorithm for finding useful macro-actions in Reinforcement Learning (Pickett and Barto[7]). The Cruncher extends PolicyBlocks by framing it in terms of ontology formation and MDL, and by adding exceptions and the ability to create multiple levels of concepts.

This paper is organized as follows: Section 2 provides a description of our representation framework and The Cruncher algorithm. Section 3 reports the results of applying The Cruncher to a variety of domains including the UCI Zoo Database and macro-action formation in Reinforcement Learning. Section 4 discusses strengths and weaknesses of The Cruncher in light of these experiments, and suggests future research directions to address The Cruncher's weaknesses.

2 The Cruncher

Given a collection of sets of attribute-value pairs, where each attribute's value is from a finite alphabet, The Cruncher produces a concept ontology which uses inheritance to compress the collection of attribute sets. One can "flatten" this ontology by computing the inheritance of every node in it, and this flattened ontology will contain the original collection of attribute sets. The Cruncher uses a greedy approach for reducing the description length of the ontology (which is

initially just the collection of attribute sets). The description length is defined as the number of links in the ontology, be they "is-a" or "has-a" links, where an "is-a" link designates that one node inherits from another, and a "has-a" link specifies an attribute that a node has and that attribute's value. (Whether the number of nodes was also included in the description length did not significantly affect our results partly because this number usually closely corresponds with the number of links.) The Cruncher generates candidate concepts by finding the "intersections" of subsets of the current items in the ontology. These candidates are then evaluated by determining the reduction in description length if each were to be adopted as concepts in the ontology. If no candidate reduces the description length, then The Cruncher halts. If a candidate is selected, then it is added to the ontology, and all other concepts in the ontology inherit from it if it reduces their description length. If there is a contradiction in the value assigned to an attribute by the nodes from which a concept inherits, that term is simply discarded. Furthermore, if a concept has an attribute, but the node from which it inherits has a different value for that attribute, then the concept states what its value is for that attribute. Thus, exceptions are allowed.

The runtime of this algorithm depends on whether one generates all possible candidate concepts, which, theoretically, can be exponential in the number of sets of original attribute pairs. In practice, one can successfully generate ontologies by randomly generating only a subset of these candidates, thus yielding a polynomial time algorithm. There is also an incremental version of this algorithm which works works by inserting one new concept at a time, and generating candidates by intersecting that node with each of the other concepts in the ontology. The top candidate is selected (or none if no candidate yields a decrease in description length), then this candidate is inserted into the ontology following the same procedure.

3 Experiments

To test the general applicability of The Cruncher, we chose a diverse set of domains to which we applied our algorithm. The UCI Zoo Database is useful for gaining insight about The Cruncher's ontologies. PolicyBlocks provides a non-bitvector domain for which there is a definite performance measure (cumulative reward). The Circles world demonstrates The Cruncher's utility for a basic sensor time-series domain. The Concept Lattice Comparison provides an example of a typical concept lattice ontology and The Cruncher's ontology on the same data set.

3.1 The UCI Zoo Database

We applied The Cruncher to the Zoo Database from the UCI Machine Learning Repository[1]. This data has 101 animals which are represented as bit vectors of length 16 (we changed the integer attribute "number of legs" to the binary "has legs" for consistency). Additionally, this dataset has a classification for each animal, which we discarded. The ontology that was created by The Cruncher

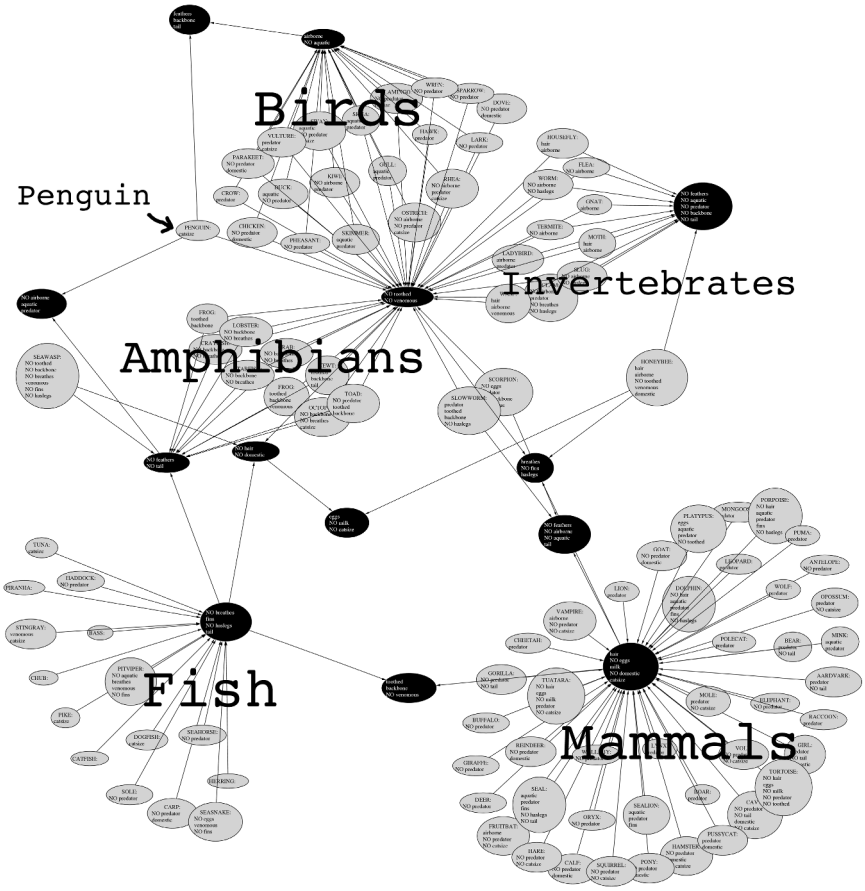

Fig. 1. The Automatically Created Zoo Ontology. Arrows represent inheritance, and the attributes and their values are listed in the nodes. Grey nodes are the original concepts in the database, and black nodes were created by The Cruncher. Mammals are grouped in the lower right, fish in the lower left, and the other three major clusters, from left to right, are, for the most part, reptiles/amphibians, birds, and invertebrates. Note, the multiply inheriting Penguin in the upper left. The class of birds is divided just for the Penguin, and the Penguin shares traits with the aquatic animals ("not airborne", "aquatic", "predator")

is shown in Figure 1. An interesting outcome is that the animals are arranged according to their classification even though this classification data was never provided. For example, there is almost a one to one correspondence to the animals that inherit from the black node in the lower right and the class Mammal. (The Tuatara and the Tortoise are the only exceptions.) Birds, Fish, and Invertebrates are likewise grouped together. The utility of allowing exceptions is demonstrated by the Platypus, which is classified as an egg-laying mammal (even though mammals are asserted as not laying eggs). The utility of having multiple

inheritance is demonstrated by the case of the Penguin, which shares traits with both aquatic life and birds.

3.2 PolicyBlocks: Creating Useful Macro-actions

In Pickett and Barto[7], it was demonstrated that for policies in a Markov Decision Process, certain concepts, which are effectively those created in the first level of abstraction in The Cruncher, can be used as useful macro-actions. These macro-actions outperform hand-chosen macro-actions such as getting to the "doorways" of the rooms in a grid-world. We started with a 20 by 20 grid-world structure (see Figure 2), and produced full policies leading to each of 20 randomly selected goal states. These policies are represented as a collection of 400 attribute-values, where the attributes are each of the 400 states, and the values are one of *up*, *down*, *left*, and *right*.

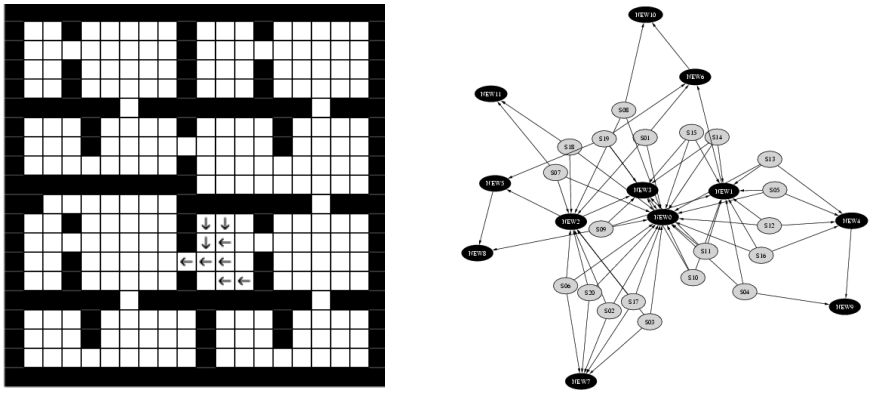

Fig. 2. A macro-action ontology. On the left is a macro-action generated by applying The Cruncher to a set of policies on a grid-world. The structure of the ontology is shown on the right. Each grey node corresponds to a full policy over the grid-world. Each black node is a sub-policy, or macro-action, that was produced by The Cruncher. For example, the macro-action encoded by the bottommost node in this ontology is that shown in the grid-world on the left. The arrows are "is-a" links, so every full policy can be thought of as the composition of all the sub-policies from which it inherits (in addition to the grey node's own modifications). Thus, each black node can be thought of as a "building block" for a full policy. (This is the origin of the term "PolicyBlocks")

3.3 The Circles World

We applied The Cruncher to the Circles domain. This is a simple 2 Dimensional physical simulation where the "particles" are circles that are "gravitationally" attracted to each other. The circles are on a torus-shaped world (i.e., the top and bottom "wrap around" to each other as do the left and right sides). There

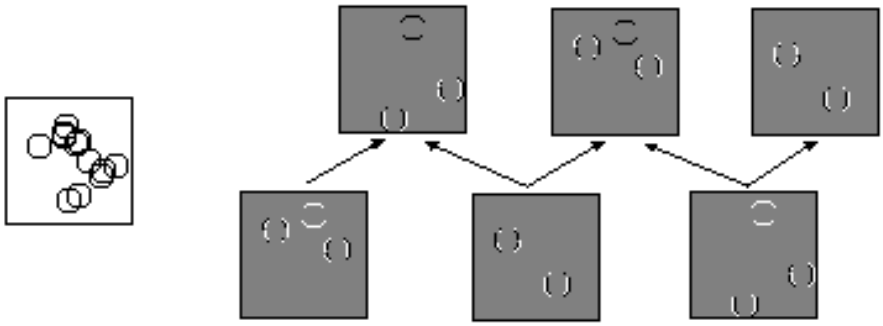

Fig. 3. The Circles World On the left is a snapshot of The Circles World, which is represented as a set of 2,500 (50 by 50) features corresponding to each of the 50 by 50 pixels. The 12 circles shown are in the process of orbiting each other in their gravitational dance. One the right is the circles top-level concept ontology. The grey areas are unspecified, and the lower 3 squares show only their *hasA* sets (as opposed to being flattened). The edges are "is-a" links. Thus, one can inherit from these "Circles concepts" to help compose a full Circles snapshot, such as the one shown on the left, just as one can use the "Policy blocks" in Figure 2 to compose full policies

are no collisions, but when the circles are sufficiently close to each other, their attraction is nullified until they are farther apart.

We provided The Cruncher with 50 "snapshots" from this simulation, where a snapshot is a set of 2,500 bits representing a 50 by 50 bitmap (see Figure 3). Note that the 2,500 bits are a raw data set. That is, no organization was provided to The Cruncher about whether, for example, Bit-1837 had anything more to do with Bit-1836 than it did with Bit-2354. This is fundamentally the same type of problem that Pierce and Kuipers[8] addressed using a different method based on statistical analysis. At the level of these bits, the concept of a circle is a fairly abstract entity. Here, The Cruncher has taken some steps toward describing the notion of a circle in that it has found that bits that form a circular pattern (when arranged in a bitmap) have something to do with each other. Figure 3 shows the top level concepts in the ontology created by The Cruncher for this domain. Noticing these correlations is the beginning of a theory of 2 Dimensional space, that is a similar result (though by different means) to the first step produced by Pierce and Kuipers[8].

3.4 Concept Lattice Comparison

The Cruncher yields better compression results than standard concept lattice layout algorithms. For example, the ontology produced by The Cruncher (Figure 4) for a Biological Organism domain had both fewer edges and fewer nodes (28 and 11, respectively) than that produced by Cole's method[3] (37 edges and 18 nodes). Additionally, formal concept lattices do not allow for exceptions, whose usefulness was shown in the case of the Platypus in the Zoo Database (see Figure 1).

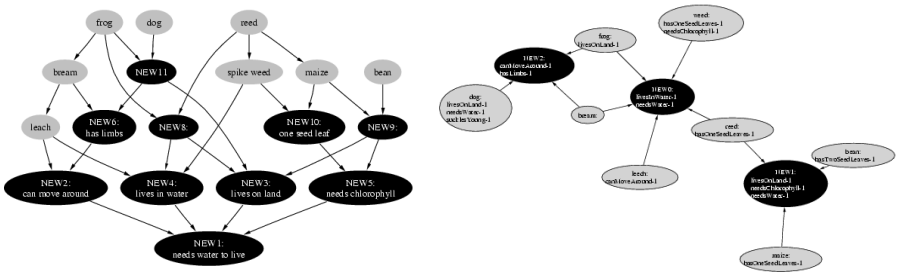

Fig. 4. Concept Lattices. A Biological Organism Domain adapted from Cole[3]. The ontology on the left was produced by a concept layout algorithm from that same paper. It has 37 edges (including the 9 "attribute" edges) and 18 nodes, which is more edges and more nodes than the ontology produced by The Cruncher shown on the right, which has 28 edges (including 17 "attribute" edges) and 11 nodes

4 Discussion

The strength of The Cruncher lies in its simplicity and its generality. The Cruncher was directly applied (i.e., with minimal massaging of the input representation) to diverse domains with positive results. Therefore, we believe that the basic ideas underlying The Cruncher may play a pivotal role in abstraction algorithms in general. The principle of MDL may be a part of a more general principle of Balance of Computational Resources. Occasionally, it is useful to cache the result of an inference, thereby trading memory for time. For example, one needs only Euclid's 5 postulates and an inference mechanism to produce all of Euclidean geometry. In practice, it's often useful to "cache" theorems rather than rederiving them even though this results in a larger description length. This resource balance may be viewed as finding Pareto optima in model space where models are evaluated by their time and memory requirements, and their accuracy. Alternatively, a model may be given a score based on some "exchange rate" among these resources. There have also been arguments that MDL alone might not be sufficient to produce useful concepts (Cohen et al.[2]) since compression tends to find frequent, but not necessarily meaningful results. However, sheer frequency is not the only factor in The Cruncher, and it would be interesting to apply our algorithm to the data set used by Cohen et al.[2] for which standard compression algorithms fail to produce meaningful results.

There are several extensions that can be made to The Cruncher. Among the most immediate of these is exploiting the heterarchy of the created ontology to speed up further crunching. This might be especially useful in the incremental version. There are some forms of concept formation that people do, but The Cruncher does not handle. For example, a person can watch a bird's eye view of a simulation of highway traffic, and quickly point out traffic jams. The person could tell you where the traffic jams are, how big they are, and how fast they are moving (traffic jams tend to move in the direction opposite that of the cars in them). A traffic jam is different from a cluster of cars because, like particles in

a wave, individual cars enter and exit a traffic jam, but the traffic jam remains. The circles in The Circles domain are like traffic jams in the sense that certain pixels turn on and off, but a pattern (i.e., the circle) remains consistent. The Cruncher also has no notion of precedence or dynamics. For example, the order of the snapshots of the circles domain was discarded. If one adds the ability to represent order, relationships, and dynamics as attribute-values, then The Cruncher can be used to organize and form concepts from stories, processes, and properties.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. Paul R. Cohen, Brent Heeringa, and Niall Adams. `icdm2002.pdf`An Unsupervised Algorithm for Segmenting Categorical Time Series into Episodes. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 99–106, 2002.
3. Richard Cole. Automated layout of concept lattices using layered diagrams and additive diagrams. In *ACSC '01: Proceedings of the 24th Australasian conference on Computer science*, pages 47–53. IEEE Computer Society, 2001.
4. D. Fasulo. An analysis of recent work on clustering algorithms, 1999.
5. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, New York, 1999.
6. T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
7. M. Pickett and A. Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2002.
8. David Pierce and Benjamin Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92:169–229, 1997.
9. J. G. Wolff. Information compression by multiple alignment, unification and search as a unifying principle in computing and cognition. *Artif. Intell. Rev.*, 19(3):193–230, 2003.