

Re-presenting Representation

Chris Thornton

Cognitive and Computing Sciences
University of Sussex
Brighton
BN1 9QH
UK

Email: Christopher.Thornton@firenet.uk.com

Tel: (44)1273 678856

May 21, 2003

1 Introduction

Representation has always been an interesting issue. Now it is a hot one too. In the good, old days of good, old-fashioned AI (60s, 70s and early 80s), it was widely agreed that representation was a key issue for cognitive science. But in recent years this assumption has come under attack [1]. Now there is growing disagreement over the relevance of representation and a steadily deepening polarization of views with respect to the necessity of employing it in cognitive machinery.

The dynamics of this dialectic are complex. However, it is fairly clear that the fortunes of ‘representationism’ are closely linked to those of ‘computationism’, i.e., the view that cognition is best understood as a form of computation. Computation is essentially the processing of data by application of operators. The ease with which a particular task can be accomplished in a computer thus depends on the appropriateness of the available operators for implementing the task with the given data. Since the operators are essentially fixed, the only way of simplifying a hard task is to *change* the data encoding, i.e., to use a ‘better’ representation of the problem. Thus, those who tend to see cognition in computational terms are led to emphasize the general significance of representation.

As researchers begin to *reject* the computationist approach to cognition [2] scepticism about the relevance of representation begins to take hold. The onus is then on the representationist to defend the position; but it begins to seem as if this may be easier said than done (but see [3]). The apparent lack of any firm theoretical foundation reduces representationism to the level of a pre-theoretical assumption. And as such it is easily written off as just another bit

of ‘computationist baggage’.

The aim of the present paper is to show that there is at least *one* firm, theoretical foundation for the representationist position. The basis for it is to be found in recent work in Machine Learning on the process known as Constructive Induction (essentially, the automatic construction of representational structure [4]). The need for such learning mechanisms has long been recognized but recent work has been able to provide a much clearer analysis of the basis of this need. This analysis provides us with the required foundational story about how learning relates to model-building and why representation has been, and will necessarily remain a key issue in cognitive science.

2 Constructive induction

Constructive induction (CI) is required when the initial representation for a learning problem obstructs the application of ordinary learning (induction) methods [4]. Wnek and Michalski [5] have divided constructive induction methods into several types including hypothesis-driven (HCI) methods, data-driven (DCI) methods and knowledge-driven (KCI) methods. Practical methods introduced in recent years include FRINGE [6], AQ17-HCI [5] and CN2-MCI [7]. Almost all CI methods seek to transform the initial representation space by introducing new features — a ‘feature’ being any mechanism which imposes a non-local partitioning on the current representation space [Thornton, Unsupervised Constructive Learning, Forthcoming]. The need for such operations is best demonstrated through a task analysis of the learning problem [8].

It is widely accepted that any learning problem can be defined in terms of a (possibly notional) target, input/output mapping. The learner’s goal is viewed as the mapping of any input taken from the target mapping onto its associated output. If the learner is to have any chance of achieving this goal, it must receive some sort of ‘feedback’, i.e., information about what the target mapping contains. In the degenerate case where the feedback provided is actually an explicit listing of the *entire* target mapping, the acquisition task is reduced to a memory task. In a non-degenerate variant of this case the learner is given access to a part of the mapping and this effectively necessitates generalization. If the access is explicit (i.e., if input/output examples are presented) we have the familiar supervised learning scenario. If access is only obtained implicitly, i.e., via some reward function, then the scenario is a form of reinforcement learning.¹

Given this formalization of learning, a task analysis can be derived by enumerating the ways in which the learner’s output guesses may be *justified* by the feedback source [Thornton, Unsupervised Constructive Learning, Forthcoming]. The feedback is, of course, ultimately derived from the target mapping. Thus guesses that are justified by the feedback must, at a deeper level, be justified by the contents of the target mapping. The guess that y is the correct output

¹Typically, reinforcement learning scenarios impose additional constraints on the availability of rewards.

for some input x , i.e., that the probability of y is 1 is therefore *only* justified if, in the target mapping, it is the case that

- (1) $P(y) = 1$, i.e., the unconditional probability of seeing output y is 1, or if
- (2) $P(y|x') = 1$, where x' is either the current input or some part of it, or if
- (3) $P(y|g(x)) = 1$, where g is some arbitrary function and x is the current input.

This taxonomy is derived simply by enumerating the possible syntactic forms for the justification. It is invariant with respect to the probability value selected (we do not have to use $p = 1$). It is also *exhaustive* since there is no alternative way of characterizing the conditional or unconditional probability of y being the right output for input x . The interesting consequence of this is that any learning method which produces justified output guesses, must exploit (i.e., use in the generation of guesses) some combination of these three forms of justification.

The cash value of this becomes clear when we consider the ways in which each of the three forms can be exploited. Exploiting a justification in this context means ‘finding’ the probability in a given distribution. Thus the complexity of exploiting a particular justification is related to the size of the relevant distribution. This prompts us to split the justification forms up according to whether the relevant distribution is *finite*. In particular, we must distinguish between what I call the **direct** forms $P(y)$ and $P(y|x)$ which are associated with finite probability distributions, and the **indirect** form $P(y|g(x))$ which is associated with an infinite one. The distribution $P(y|g(x))$ is infinite due to the infinite number of choices to be made regarding g , which is defined as any computable function.

This analysis of justification sources leads directly to a fundamental insight concerning the complexity of learning problems. Problems which involve exploiting either of the two direct forms involve the equivalent of sampling a finite distribution while problems which involve exploiting the indirect form involve the equivalent of sampling an infinite distribution. Other things being equal, the task of exploiting direct forms thus has a *lower* theoretical complexity than the task of exploiting the indirect form. Note that this is a qualitative distinction similar to the one between polynomial and exponential time complexity. However, it has a firm, mathematical basis in the enumeration of syntactic forms for probability values.

2.1 Statistical v. relational problems

It is important to note that values of the function g (which I call the **recoding function** below) must depend on *relative* argument values. In other words, the function must compute or evaluate a relational property of its arguments. Were we to have an indirect justification in which values of the recoding function depended on the *absolute* values of its arguments, then we could specify the same justification purely in terms of those absolute values, i.e., by deleting the

g and associated parentheses. Thus, all indirect-form justifications necessarily involve functions which effectively compute relationships of their arguments.

In recognition of this, I call the class of higher-complexity problems which involve exploiting indirect justifications **relational** and the class of lower-complexity, direct-justification problems **statistical** (since exploiting a direct-form justification involves sampling for a statistical effect in the form of an observed probability). If it was not obvious before, the introduction of this new terminology should drive home the point that this analysis gives a foundation to the well-known Machine Learning heuristic which states that ‘learning relationships is hard’ [9].

2.2 Relevance to unsupervised learning

Although the analysis takes the idea of a target input/output mapping as a starting point, it is not, in any way, restricted to the supervised learning scenario. In unsupervised learning, the aim is to model the input environment, i.e., to find its ‘regularities’. The successful achievement of this goal must facilitate prediction, i.e. the guessing of correct values for input variables. Thus we can regard unsupervised learning as a form of supervised learning in which one set of variables serves for both input and output, and where it is not known in advance which variables are to hold the outputs. Since the analysis makes no assumptions about *prior* distinctions between input and output variables, it applies equally well to the unsupervised as to the supervised scenario. Thus we can apply the exploitation rule given above to all forms of empirical learning. To summarize

- Empirical learning of any form necessarily involves the exploitation of either statistical or relational effects, or some combination of the two.

2.3 ‘Constructive’ means ‘relation-exploiting’

The task analysis allows a clean distinction to be made between constructive and non-constructive forms of learning. The essence of the distinction is that constructive learning can be equated with exploitation of relational effects while non-constructive learning can be equated with exploitation of statistical effects. Defining the distinction this way preserves the crucial notion that constructive learning involves creating features which implement complex, ‘non-local’ partitions of the underlying representation space.

Any feature/function which computes a statistical property of its arguments will tend to define a partitioning involving contiguous regions of the original input space, whereas a feature which computes a relational property will not do so. The argument is as follows. The values of a function which computes a statistical property of its inputs are, by definition, correlated with the absolute values of the inputs to that function. Thus values of the function are correlated with absolute ‘coordinates’ of the representation space and the function must,

therefore, define sets of objects which tend to share the same coordinates, i.e., be located within the same region of that space.

In contrast, the values of a function that computes a relational property of its inputs are not correlated with the absolute values of the function's inputs and thus not correlated with absolute 'coordinates' of the representation space. The feature implemented by the function will therefore tend to instantiate a complex, non-local partition of the space.

There thus appears to be a good fit between the relational/statistical distinction and the constructive/non-constructive distinction. (This has been implicitly recognized by a number of authors, e.g., [10]). It is justifiable, then, to ground the notion of constructive learning in the mathematically precise notion of relational exploitation. To summarize

- Constructive learning is the exploitation of relational effects.

3 Making what is implicit explicit

The task analysis shows that there are essentially two forms of learning: an easy form which involves exploiting statistical effects and a hard form which involves exploiting relational effects. It also shows that a relational effect is necessarily exploited via the identification of a recoding function or 'feature'. The feature effectively creates a new, 1-d representation subspace in which the effect in question is re-expressed in a statistical form.

These conclusions turn out to have interesting implications for the issue of representation. Note that the statistical effects observed in some data are *explicit* properties of those data. They exist on the surface. They are not, in any sense, hidden. They can be detected/measured using the simplest form of machinery, e.g., a single artificial neuron or a single conjunctive formula. In contrast, relational effects are *implicit* properties of the data. They are hidden beneath the surface and can only be brought to light via the application of a particular function (or feature). Thus the constructive/non-constructive distinction effectively picks out the difference between learning about *explicit* properties of the input environment and learning about *implicit* properties of that environment.

A learning agent that exploits a relational effect 'captures' an implicit property of its input environment. The capture is embodied in a recoding function and this forms what is in effect a virtual sensor for the implicit property. The agent thus expands its own sensory (input) environment. The result may be that further implicit properties become salient and worth capturing.

To illustrate the nature of this process, imagine a simple robot that uses a number of range-finder sensors. These measure the range to the nearest, facing surface in a given direction. The presence of something large enough to count as an obstruction is not an explicit (statistical) property of the robot's sensory environment. Rather, it is an implicit property: an 'obstacle' is detected if several, closely-aligned sensors show similar range values. If the robot operates

as a constructive learner, it can be expected to exploit this relational effect and to thereby capture the implicit property of ‘obstacleness’.

In taking this action, the agent acquires a virtual sensor for ‘obstacleness’. Following this, the property of obstacle size may become salient. Again, this is not an explicit property of the sensory environment but *is* a relational property of the expanded environment since it can be derived from the ratio between the apparent width of an obstacle (number of range finders showing same range) and its apparent distance (range shown). The capturing of ‘obstacle size’ expands the sensory environment further and opens up further virtual-sensory possibilities. For example, if the agent has (or acquires) some measurement of time, then implicit temporal properties such as obstacle velocity and momentum may become salient.

The key point here is that constructive induction is a form of bootstrapping process. Each constructive step puts into place virtual sensors for properties which are implicit with respect to the current sensory environment. Thus the process incrementally builds a recoding framework which ‘models’ emergence structures in the environment. The abstract property of momentum, for example, is an emergent property of size and velocity. The constructive learner that produces features (recoding functions) which measure size and momentum is thus engaged in the process of modeling a structure which exists in the environment. The internal implementation of these recoding structures are thus *literally* representations (to the agent) of external structures.²

The task analysis of learning shows, then, that learning which seeks to move beyond the capture of simple statistical effects is necessarily constructive. Each constructive step turns a relational effect into a statistical effect and thus transforms an implicit property of the input environment into an explicit property of an *expanded* input environment. The general conclusion is that learners which operate in complex environments must incrementally construct internal representations of the structures which exist in those environments.

The take-home message should be clear: learning is either *trivial* or it is *representational*. Note that this is not speculation; it is a deduction from the task analysis. It does not establish a link between cognition and representation. But it does establish a link between *learning* and cognition. And to the extent that cognition is assumed to be part and parcel with learning, it therefore legitimates a modest representationist position.

4 Step signatures

As I noted above, constructive learning in its most general form is *hard*. At each stage, it involves the discovery of a particular feature (recoding function);

²The fact that these representations are *within* the agent, and thus a part of it, should not confuse us here. In saying they are representations ‘to the agent’ we implicitly refer to the agent that remains when the representations are relocated in an imaginary external environment. We employ the same trick when we talk about sensors providing information ‘to the agent’.

this may be any computable function and must therefore be selected from the space of *all* computable functions. In the worst case, then, constructive learning involves multiple searches through an infinite space. Fortunately, there are ways in which the effects of constructive learning can be achieved without invoking any sort of search whatsoever.

Note that it is possible to have a relational effect in which the roles of the relevant relationship (or the values which the roles take in each case) always instantiate the same variables. For example, consider again the range-finding mobot. Imagine that some obstacles in the environment have aggressive intentions. These obstacles tend to attack the mobot from the front, rotating as they move along the reciprocal of the mobot's own trajectory. The mobot has no sensor which directly tests for 'attacking, rotating obstacle'. However, the presence of such an obstacle is an implicit property of the current sensory field, i.e., it is a relational effect. Provided the attack always comes from the front, only those sensory inputs which are derived from the sensors which point ahead will participate in the effect. Thus the relational effect will operate over a small, *fixed* collection of inputs.

Relational effects such as these, which always instantiate the same input variables, may create reliable statistical 'signatures'. If we take a large number of examples of the effect and sample for statistic effects, the similarity groupings that emerge constitute statistically typical cases of the underlying relationship. They can therefore be treated as prototypes for that relationship. An approximation of the value of the relevant relational function (as applied to an unseen case can) then be derived simply by comparing the unseen case against the prototypes and returning the value associated with the best match (see [Thornton, Unsupervised Constructive Learning, Forthcoming] for details).

An example may clarify the way in which this function-approximation process works. Consider the training cases listed below. Each line here contains an input/output pair (input on the left of the arrow, output on the right). The underlying rule is that the output value is simply the difference of the input values. Any given value of either of the two input variables may be associated with any value of the output variable.³ Thus in a suitably exhaustive version of this training set we will expect to see negligible statistical effects.

0.17	0.48	-->	0.31
0.19	0.50	-->	0.31
0.65	0.82	-->	0.16
0.59	0.76	-->	0.16
0.96	0.18	-->	0.78
0.99	0.21	-->	0.78
0.45	0.48	-->	0.03
0.44	0.47	-->	0.03
0.88	0.45	-->	0.43
0.84	0.41	-->	0.43
0.86	0.43	-->	0.43

³'Clipping' effects withstanding.

However, note that the relational effect creates a fairly obvious statistical signature. This is nicely illustrated by showing the input cases as points in the 2-dimensional input space (see Figure 1). Note how the points (input cases) group themselves into four main clusters, picked out by dashed circles. If we identify these groupings and then arrange them into an ordering (see the thick, double-headed line) we have what is, in effect, an approximate recovery of the underlying relational effect. Presented with an arbitrary input we can see which point it instantiates in the 2-d space, how that point is situated with respect to the relational dimension and then simply return the output associated with the nearest cluster (prototype). The double-headed line, then, can be thought of as an approximate recovery of the underlying relational ‘dimension’. For instance,

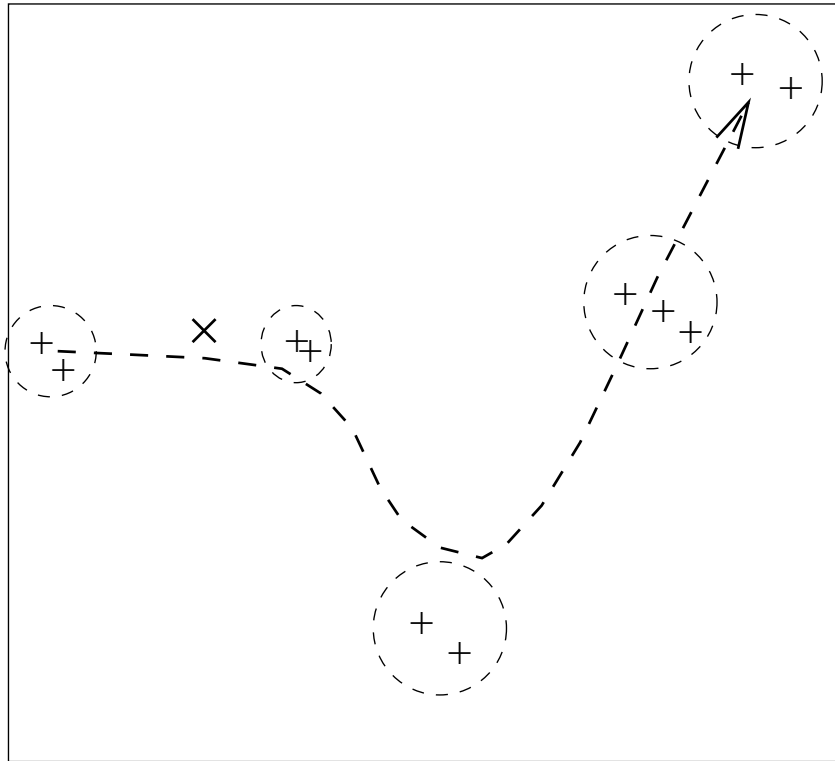


Figure 1:

consider an input such as

0.20 0.05

No such input appears in the training examples. However, it is easily seen that it instantiates a point in the 2-d input space that is about half-way between the two left-hand clusters. The mean outputs for these are 0.31 and 0.03. Thus

interpolation gives us an output of 0.14 as the output for $\langle 0.20 \ 0.05 \rangle$, which is very close to the correct output of 0.15. Interestingly, this is obtained without there being any real recovery or implementation of the underlying differencing function.

Of course, the accuracy of the recovery achieved by this method all depends on the clarity of the signature. If the statistical clumps pick out a uniform, densely instantiated range of relational cases then the recovery will be relatively accurate. If the clumps are weakly defined, or are scattered haphazardly through the space, the prototypes will be noisy and the recovery poor.

4.1 Explication

I call relational effects which create prototype-yielding signatures ‘step effects’ or ‘step signatures’. Their existence in an environment lays down a royal road for the short-winded constructive learner. To follow it the learner must continually expand its input environment through the prototype-exploiting process described above. But each step in this process involves nothing more than ordinary statistical sampling. Explicit search of the space of computable functions is never required. I call this process ‘step explication’, or just ‘explication’ for short, on the grounds that it incrementally renders implicit properties of the current input environment explicit. I have carried out various investigations into properties of this algorithm and the results are written up elsewhere [Thornton, Unsupervised Constructive Learning, Forthcoming] [11].

5 Discussion

Traditionally, AI research has adopted a firmly representationist position on cognition. However, the fact that representationism was never given a proper foundation has made the propagation of misleading caricatures of it only too easy. In recent years, the representationist has sometimes been portrayed as one who believes that cognition necessarily involves the forced mediation of sensory input through faithfully accurate, complete-in-every-detail re-renderings of user-specified external worlds [1]. This is not a particularly accurate characterization of in-practice representationism. But representationists have only themselves to blame for the emergence of such ‘straw men’.

In the present paper I have shown that the task analysis of learning clearly differentiates two variants of the learning process, an easy form which involves exploitation of statistical effects and a hard variant which involves exploitation of relational effects. The latter form of learning, which is equated with the process of constructive induction, has been shown to involve the making explicit of implicit properties of the environment. Where learners operate in complex, structured environments the constructive process necessarily attempts to recover those structures which are behaviourally salient. In other words, it engages in the attempt to construct representational models of the behaviourally salient aspects of the environment.