WORKING NOTE AC123

Alan Creak
1998 June 10

LEARNING TO LEARN : A COMPARISON

Paul Qualtrough[1] asserts that if machines - particularly robots, but I don't think that's significant here - are to be able to perform useful functions in the complexities of the world as it is, then they must be able to learn for themselves. After prodigious efforts, Paul has succeeded in constructing a machine which can find its way reliably from one end of a straight line to the other. I don't think it has a name, so I shall refer to it (and to any of its close relations) as PQ.

Maulsby and Witten[2] report on a machine they call Cima, "an interactive machine learning system ... Designed to acquire behavior concepts from a few examples ...". So far as I can determine from the paper, it too operates with limited sensory equipment, though as well as observing the immediate environment within a text, it can perceive hints presented by a teacher. (PQ is also intended to accept hints, and other forms of cooperation, in due course; and Cima can operate without the hints.) Cima has no understanding of its activities. It operates by associating a sort of clump with each significant structure it learns to observe; unlike PQ, Cima works on one clump at a time, learning every modification directly from its teacher, and producing clumps which can in principle become very elaborate. Once made, though, these clumps are not used for anything else - indeed, Cima develops no knowledge on its own initiative. This machine can learn simple editing operations after a few exchanges with its teacher, and then apply these reliably throughout a document.

There are certainly basic similarities between the two approaches, but the level of performance is vastly different; PQ struggles to learn a trivially simple task, while Cima acquires new skills in editing operations with impressive facility and speed. It is interesting to attempt a comparison in order to determine where the difference lies.

THERE ARE REALLY THREE METHODS.

Maulsby and Witten describe two sorts of experiment : first a "Wizard of Oz" experiment (Turvy), conducted to learn about the requirements, then a real Lisp implementation (Cima). The same general principles apply in both cases, but the abilities of the two systems are noticeably different, though the precise nature of the difference is not defined. It is stated that in the Turvy experiment "we limited the types of instructions (sic) Turvy could understand" (page 37), but the limitations are not presented. Turvy certainly deals with hints in a more sophisticated way : "Cima merely spots keywords; Turvy parses sentences" (page 43).The examples described suggest that Turvy is indeed somewhat more clever than Cima, so I shall regard them as different methods. And PQ makes three.

Despite the difference, I don't think that there's enough detail in the paper to distinguish between Turvy and Cima in any productive way here. I've left these comments here because I think they are at least slightly significant, and a more complete analysis should include both methods. The more complete analysis should also take into account information which I don't have about the details of the three methods. I infer that the more complete analysis is unlikely ever to be done, by me anyway.

INSTANT COMPARISON.

PQ begins with very limited sensory equipment and no knowledge of their environment, and by trial and error learn how best to respond to their currently observed environments and recent history. The responses are composed from a set of actions which initially comprises those actions built into the machine; the machine has no model with which it can relate the actions to its perceptions, and must learn the effects of the actions from scratch, with the help of feedback from the environment of the "cost" (positive or negative) of each action, after the event. As it gains experience, it might notice that certain sequences of elementary actions are commonly useful, and it can then compose these into a single action by "clumping". The composite action then becomes available as a unit which can be used in other contexts if should prove of value.

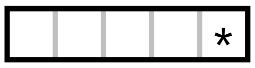
Both Cima and Turvy begin with a great deal of world knowledge about the task to be accomplished. In the task described in the paper, Turvy understands quite a lot of English, including parsing and punctuation, and it knows what sorts of thing are important attributes of the objects it sees - such as the distinction between letters and words, the significance of capital letters, that brackets are special things, etc. It knows that there is an authoritative teacher who always speaks the truth (though it might not be the whole truth), and it begins by inferring an algorithm from a worked example provided by the teacher. The algorithm is refined by more examples, or by the system performing what it believes to be the algorithm until stopped by the teacher at a mistake, when it can try to guess at an extension of its rule, or accept hints from the teacher. The Cima task described is generally similar, adding little to the principles involved, but it does demonstrate that the task really can be automated.

PAUL'S STRAIGHT LINE.

A brief description of Paul's straight line problem, successfully solved by PQ, is presented so I know what I'm talking about when I refer back to this (if ever).

In this exercise, PQ lives in a two-dimensional world of square cells. Certain inter-cell boundaries have walls; others don't. PQ's task is to move from its current position to an identified goal. It doesn't know that it has a task or a goal; its only information is that if it reaches the position which *we* know is a goal, then it receives a considerable reward of some useful resource, and that every movement costs it some resource. Trying to walk through a wall costs it a lot of resource. The agent's view of the world is limited to the edges of its current cell (so it can see the walls, if there are any), and its perception of its current store of resource. Its possible actions are to move North, South, East, or West. It has a memory, which remembers a selection of its history as a sequence of { cell type, move chosen, cost } triplets.

In the straight-line problem, the agent lives in a world like this :



The goal is marked by the asterisk. From the agent's point of view, there are three distinguishable places, identifiable by the presence or absence of walls in the N, E, S, and W directions as 1011, 1010, and 1110; it must learn that the best strategy is to move East in the first two cases, and something else (irrelevant for present purposes) in the third.

COMPARING THE METHODS.

I have compared the methods by gedanken experiments in which I consider the solution of problems solved by PQ, by Cima, and by Turvy using methods which I hope resemble those used by Paul and by Maulsby and Witten. I do not even try to provide real solutions; my intention is only to point out the differences between the different methods.

PQ and the straight-line problem.

PQ (like Cima and Turvy) works by associating actions with records of its perceptions. It uses a reinforcement-learning method. It begins with no knowledge. In any position, it chooses an action on the basis of what it remembers of the consequences of previous actions taken with the same perceptions; if some action turns out to be good in terms of resource costs, the memory is adapted accordingly. After quite some time, it succeeds in learning the expected strategy.

Cima or Turvy and the straight-line problem.

Hint 1 : If the world looks like 1011, go East.
Hint 2 : If the world looks like 1010, go East.
Hint 3 : If the world looks like 1110, do something else.

Problem solved.

PQ and the bibliography problem.

The first question which must be asked is whether PQ could in principle solve the bibliography problem. The answer is almost certainly "yes". PQ is so designed that it has no intrinsic "understanding" of either its perception or its actions, and it proceeds by seeking relationships between whatever symbols it perceives and the actions which it takes. A version of PQ constructed to work on the bibliography problem would be able (like the version for the straight line problem) to move along a line, which is the character string, to left and right, and to sense the characters one by one. It would be endowed with a new action, the ability to write into the character string. I *think* that's all it would need, though it might be of assistance to add actions to copy the current character and write the copied character; without these, it would have to learn separate operation sequences for each different character it wanted to move.

Given this new sort of PQ, then, consider the example on page 38.

It begins with a hint : "Take the last name, before the colon", accompanied by the sequence of actions "Select 'Agre', copy, move to start of paragraph, make new paragraph, type '[', paste, type ']', set style 'citation'". (The sequence might not be precisely correct, but it's about right; I derived my guess from various clues in the paper.)

To make sense of this, PQ would have to perceive the hint and the selection. Its perception of the context of the selection must include these points :

- 'Agre' is a name. This is necessary to match the selection to the description.
- 'Agre' is followed by a colon. Also necessary to interpret the hint.
- It must know that the character ':' is a colon.

PQ must also know that :

- the selection of an explicit text item accompanied by a more general description given as a hint identifies a pattern of behaviour which is to be applied to any text satisfying the more general description. (Though why doesn't that work with "the last two digits of the date" in the second hint ?)
- Unless there is an explicit "move to the start of this paragraph" instruction, it must be able to identify the insertion point of the new paragraph as the beginning of the same paragraph as that including 'Agre' - perhaps by inspecting the parse tree ?

These items of knowledge are more to do with procedure than perception.

Proceeding through the rest of the sequence produces further examples, generally of a similar nature; it is clear that Turvy's perception of its environment is assisted by a great deal of analysis - so, for example, it knows not only the immediate item under inspection but also its context, and it knows whether the item is a word or a letter or a number or a punctuation mark, and it knows whether a word is expressed in upper or lower case letters. There is also quite a lot of special procedure built in.

Are there any conclusions for PQ ? A version of PQ incorporating all these attributes as the perceptions of specialised sensory organs could be built - but the result would be an agent specialised for work of a very limited sort. Special procedures could not obviously be incorporated without doing violence to the principles of PQ.

To preserve the essence of PQ, a more appropriate approach would be to encourage it to learn to distinguish digits from letters, to parse (at least at the level of lexical analysis), and so on. I don't see any reason why it couldn't do that, though again an extension is suggested. At present, it can learn to classify its experience in terms of { move, percept } pairs of the forms { M, P }, { *, P }, and { M, * }, where * is a wild character. To form a notion of digits, it would help if an intermediate level of classification of the form { M, { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 } } could be identified. It is true that in principle this is only a matter of convenience, as without the extension it is only necessary for the machine to learn the cases separately - but to cope with the telephone number example some means of coping with all four-digit numbers is required, and it would take some time to learn the requisite 10000 rules.

Nevertheless, it is interesting to ask whether the prowess of PQ could be demonstrated on a problem in some sense equivalent to the bibliography problem. I think such a simplified problem can be defined. The bibliography problem has three significant components :

1 : Identify a significant string;
2 : Go to an appropriate place;
3 : Perform some defined action.

Here is a simple problem which follows this pattern : in a string such as that shown below, find the sequence of two 3 symbols, then proceed to the left until a 1 symbol is found, then ring a bell. A suitable string would be

```
1111111222222222222222331111111
```

Only the three percepts 1, 2, and 3 need be distinguished; and the only actions are left, right, and bell. The bell could also reasonably include the generation of a new string and a teleport into it somewhere. I think some internal state might also be necessary. Could one argue that a demonstration that a PQ could solve this problem would show that it was as capable as the Maulsby and Witten system ?

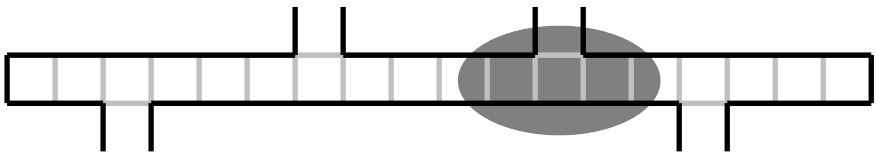
A SOLVABLE PROBLEM ?

Masliah and Albrecht[3] solve robot navigation problems by getting people to do them using only the input available to the robot. Their example is a mobile robot navigated by sonar along a corridor. Given only the sonar inputs displayed on a screen, the people were able to find doorways in the corridor, and navigate the robots through them - though they couldn't determine whether or not the doors were open, which seems odd.

Their next step was to use cues from the people's way of addressing the problem to show how to programme the robots to do the same job, which they did, though not as well as the people. This makes Paul's point very well : coping with the real world is too hard to programme. It doesn't go on to the next step, which is to make the robot do the learning.

Can PQ do that ? It seems to be exactly the right sort of problem. I think it could.

The solution depends on noticing the special sorts of sonar reflection from the doorways; so consider this simplified form of the problem :



Again, I think I've eliminated incidentals and kept the essence. The dark grey ellipse shows the limits of the robot's vision, and that can be encoded as a set of eight tokens, each indicating the type of the corresponding piece of wall. Examples might be :

@	+	+	+
@	+	*	*

+	*	*	+
+	+	+	+

+	+	+	*
+	+	+	+

(Each of those appears in the diagram; work it out. As the diagram stands, there's no point at which the robot sees nothing but plain wall, and that's perhaps not realistic, but I'm too idle to change it.) The robot gets a big reward if it moves north or south through a doorway.

The next question is : How could you hook together a PQ and one of the people whom Masliah and Albrecht refer to as Mobile Robot Surrogates (MRS - I hope that's why they call them "she") so that the actions of the MRS are interpreted as hints by the PQ ?

BRAINSTORM

Consider the hypothetical PQ proposed for the "solution" of the bibliography problem. Extend it just a little in the direction of further compliance with the conditions of the real problem by adding actions which will allow it to write 1, 2, or 3 at its current position in the string.

Now it can move along a string, inspect and notice the characters it sees, and alter them; it has all the machinery necessary - when seen from the outside - to learn to be a Turing machine. Is it possible ? From the inside, it doesn't have anything corresponding obviously to a state - though its memory might be equivalent. Or might not. Or one could be added. Or learnt ?

Does that mean that a PQ could, in principle, learn to compute any computable function ? I don't know, but it sounds like a good trick. Could it learn to be a universal Turing machine ? - that's an interesting question, because you can build astonishingly simple universal Turing machines with a very small number of internal states, which means that you wouldn't have to incorporate an autonomous state-generating action which might run wild.

Should the PQ have provision for state ? It's a way of being able to take different actions in specific circumstances, depending on what you're doing at the time, so it offers an escape from perceptual aliasing.

DISCUSSION.

This comparison can be interpreted as showing *either* how much you have to provide, and how restricted the result is, if you want to make an efficient system, *or* what a lot there is to learn, and how hard it is, if you want the system to learn for itself.

A major difference is in the amount of structural knowledge available to the agents in the different cases. Cima and Turvy are on much the same level; they have built-in knowledge of the differences between letters, numbers, and punctuation characters, of words, of upper case and lower case, of bracketing, of the significance of the case of the first character of a word, and much more. (Some examples of Cima's knowledge are presented on page 41 - and, from page 40, Cima's knowledge "includes facts about data types, generalization hierarchies, methods for matching and generalizing examples, default rankings for the salience (that is, 'interestingness') of example attributes, and directed graphs that encode suggested changes in focus of attention". That's quite a lot.) PQ knows exactly none of this, and must learn it all by inference from the examples it receives. Cima and Turvy also know the importance of hints; PQ doesn't, and perceives a hint only as additional sensory input to be evaluated along with the rest. Many other similar comparisons could doubtless be suggested, with the common feature that almost everything built into Cima and Turvy must be learned by PQ from its experience alone.

The title chosen by Maulsby and Witten is "Teaching agents to learn : from user study to implementation". It is hard to see this as anything but a joke; there is no sense in which their agents can learn any better after their experiments than before. Instead, the agents learn because they are programmed to do so, not in any sense taught. And they're programmed in enormous detail. Cima has been tested in several different fields and has given good results; details are not presented in the paper, but one imagines that a substructure parallel to that required for the text editing must be required before the basic method can be effective.

The agents used by Maulsby and Witten are in fact taught to perform simple, and very specialised, routine tasks. Could the teaching be elaborated to the extent of really teaching them to learn ? For several reasons, this is not an easy question to answer. First, the meaning of being "taught to learn" must be defined; I have previously got myself into a tangle[4] by trying to define and distinguish between notions such as learning to do, learning to learn, learning to learn to do, and so on. The intended meaning is something like "able to acquire new learning methods through interactions of the sorts currently used in teaching tasks, and certainly without any additional programming". Second, it might very well be that the structure of the agents is constrained to the extent that it is not possible to add necessary new instructions. Third (and very like second, but I need at least three to justify "several"), even if instructions can be added, the existing machinery might not be able to handle necessarily more complex data structures. I don't propose to pursue this question further, because I don't have the information or the spare time, but it is certainly not obvious that the answer is "yes". It is instructive, in a negative sort of way, to consider what you would do to teach Turvy a new way of learning things.

Is Paul overdoing it ? I don't think so. PQ is dreadfully slow and primitive, but perhaps that is the inevitable price of generality. Can he use some of the M&W ideas to extend PQ and speed it up without losing generality ? Again, not obviously; without working through the sort of analysis which would make me certain (or even knowing what sort of analysis that is), the M&W methods seem to be such that they gain efficiency in special cases by ruling out activities which might not always be unwanted in a system designed for generality. PQ might take a long time to learn to rival Cima in the performance of finding telephone numbers in text, but the same PQ can then go on to learn to drive a car. It isn't obvious that the same Cima can do that.

Paul's methods will lead to a mode of operation in which the machines themselves learn all the material preprogrammed into the system developed by Maulsby and Witten - which is to say, all the substructure which they need in order to learn their tasks. To that extent at least, Paul is addressing the problem of learning to learn.

The modelling the MRS method experiment is interesting in that it both ties PQ to its intended arena of action - real mobile robots - and shows how hints might be interpreted. I've left out a step, I think, and that's the robot's task of connecting the hints from the MRS (which amount roughly to something rather like one of the simplified vision patterns in the experiment) with the real sensor data. That looks very like a pattern recognition problem of a sort that shouldn't be terribly hard to solve, and any appropriate self-respecting neural network should be able to do it. Said he, confidently. Anyway, the example convinces me that PQ is close enough to real to be a significant development in more than a theoretical sense.

The brainstorm probably is just that, but it suggests an interesting link. I don't think it extends much further than showing that in principle the architecture of PQ, with state added, could in principle suffice to implement a Turing machine. That doesn't say much - you can say the same for almost any computer. Unless one can demonstrate that there's a real algorithm which could learn to be a Turing machine, it doesn't get you much further - but if there *is* such an algorithm, that would be really interesting.

REFERENCES.

1 : P.T. Qualtrough : divers conversations, whiteboard drawings, thesis drafts (< 1999).

2 : D. Maulsby, I.H. Witten : "Teaching agents to learn : from user study to implementation", *IEEE Computer* **30#11**, 36-44 (November, 1997).

3 : M.R. Masliah, R.W. Albrecht : "The mobile robot surrogate method for developing autonomy", *IEEE Trans. Robotics Automation* **14**, 314-320 (1998).

4 : G.A. Creak : *Thoughts on building autonomous adapting machines*, unpublished working note AC90 (December, 1993).