Experience Selection and Problem Choice in an Exploratory Learning System

PAUL D. SCOTT

Department of Computer Science, University of Essex, Colchester CO4 3SQ, United Kingdom

SCOTPD@UK.AC.ESSEX

SHAUL MARKOVITCH

Computer Science Department, Technion, Haifa 32000, Israel

SHAULM@TECHSEL.BITNET

Abstract. A fully autonomous exploratory learning system must perform two tasks that are not required of supervised learning systems: experience selection and problem choice. Experience selection is the process of choosing informative training examples from the space of all possible examples. Problem choice is the process of identifying defects in the domain theory and determining which should be remedied next. These processes are closely related because the degree to which a specific experience is informative depends on the particular defects in the domain theory that the system is attempting to remedy. In this article we propose a general control structure for exploratory learning in which problem choice by an information-theoretic “curiosity” heuristic: the problem chosen then guides the selection of training examples. An implementation of an exploratory learning system based on this control structure is described, and a series of experimental results are presented.

Keywords. Exploratory learning, experience selection, problem choice, machine discovery, curiosity.

1. Introduction

This article is concerned with problem choice and experience selection for *exploratory learning*: that is, learning in which a system attempts to acquire a good theory of a novel domain without any form of assistance from an external teacher. We assume that such an exploratory learning system is acquiring a theory in order to solve any problems that may be encountered in the domain, but that, at the time of learning, the system has no knowledge of what these problems might be. Its goal is to produce a theory of the domain that is correct, complete, and consistent, as measured by the quality and scope of its predictions. The system is thus a scientist attempting to discover the physics of the domain rather than an engineer attempting to find ways of changing it.

1.1. Experience selection

Any learning or discovery procedure can be viewed as the interaction of two distinct search processes (Simon & Lea, 1974; Shalin, Wisniewski, Levi, & Scott, 1987). The *theory generator*¹ searches the space of possible theories for one that provides a good model of the domain that is being learned. The *experience selector* searches the space of possible experiences of that domain for a set of experiences that will make a useful contribution to the search for an improved theory. Thus the objective of the experience selector is to

choose informative experiences: that is, experiences that can be used by the learning system to remedy some defect in its current theory of the domain.

A good strategy for searching experience space can have a major impact on the effectiveness of any particular method of searching theory space. Winston (1975) drew attention to the critical role of *near miss* training examples, while more recently both theoretical analyses (Mitchell, 1982) and empirical studies (Ruff & Dietterich, 1989; Scott & Markovitch, 1989) have demonstrated that experience selection strategies that use information about the current state of the search of theory space can drastically reduce the number of training experiences required. A review of experience selection techniques used in both supervised and unsupervised learning systems can be found in Markovitch and Scott (1992).

1.2. Problem choice

Problem choice is the process of identifying defects in the domain theory and determining which should be remedied next. It is closely related to experience selection because the degree to which a specific experience is informative depends on, among other factors, the particular defects in the domain theory that the system is attempting to remedy. Almost all existing machine learning systems rely on an external agent to determine the problems upon which they will work. Exploratory learning systems have been provided with problems in two distinct ways. Systems like the BACON dynasty of programs (Langley, Simon, Bradshaw & Żytkow, 1987) are required to find laws relating a set of variables. As Langley et al. (1987) have noted, problem choice, which they term "discovery of research problems," is an important feature of scientific research that falls outside the scope of the BACON programs. Systems such as PRODIGY (Carbonell & Gil, 1987), LIVE (Shen & Simon, 1989) and COAST (Rajamoney, 1989; Rajamoney, DeJong, & Faltings, 1985) are given a set of problems to solve and a defective domain theory: defects that are discovered while attempting to solve the problems must be remedied through learning. Both these approaches require an external agent to play a role in problem choice, but a fully autonomous exploratory learning system must be able to set its own learning agenda.

2. A curiosity-driven control structure heuristic

Both experience selection and problem choice are important, and perhaps somewhat neglected, aspects of machine learning. The goal of the work described in this article is to develop an exploratory learning system that is fully autonomous because it is able to choose its own problems and select its own experiences without any form of external assistance.

2.1. The need for a curiosity heuristic

The difficulties facing an exploratory learning system that must choose its own problems are illustrated by the following fable. Consider an ignorant man who decides to educate

himself and has access to a very large library: he must devise some criteria for deciding which books to read and in which order. If he were to employ a *novelty heuristic*, always reading about the topics of which he was most ignorant, he would become a pathological generalist, discarding each book as soon as he had a slight acquaintance with its first few pages in favor of a volume on a completely different subject. On the other hand, if he were to employ a *familiarity heuristic*, always reading about the topics on which he was most informed, he would become a pathological specialist, continually reacquainting himself with information he has already acquired.

Clearly what is needed is some other selection strategy that avoids both the excessive superficiality produced by the novelty heuristic and the equally excessive narrowness produced by the familiarity heuristic. No real human being would adopt either of these extreme strategies. Such a man is much more likely to be driven to gradually expand his intellectual horizons as the material he read aroused his curiosity in one or both of two ways. He could discover some ideas or principles but be left uncertain about their range of applicability; or he could find that what he had read was ambiguous or contradictory and thus be left uncertain about what was really meant. In either case his choice of further reading will be determined by his wish to resolve the uncertainty that has arisen as a result of his initial reading.

FAHRENHEIT (Żytkow, 1987; Langley & Żytkow, 1989; Żytkow & Zhu, 1991) incorporates a problem-choosing facility that seeks to resolve uncertainty about range of applicability. After the program has produced a law, using its BACON-like module, its *scope search* facility is invoked to generate experiences outside the range of those previously considered in an attempt to find the boundaries of the region in which the law applies. Once a number of such boundary points have been discovered, they themselves can be used as experiences by the BACON module in an attempt to discover laws defining the boundaries. Further applications of the BACON module can also be used to discover alternative laws for regions outside the boundaries, thus leading eventually to a complete theory.

2.2. Uncertainty-driven search

In order to develop a heuristic that enables a system to resolve uncertainty arising from ambiguity and inconsistency, we require a method of measuring such characteristics in the current state of a domain theory. If one performs an experiment for which the possible outcomes fall into a number of distinguishable classes, the uncertainty, H , of the actual result is given by the Shannon uncertainty function (Shannon & Weaver, 1949)

$$H = - \sum p_i \ln_2 p_i \quad (1)$$

where p_i denotes the a priori probability that the outcome will belong to the i th class. A situation of complete familiarity is one in which it is known in advance to which class the outcome will belong. In such a case, only that class has a non-zero probability of occurring and hence the uncertainty is zero. In contrast, in a situation of complete novelty, one not only is ignorant of which class of outcome will occur but also lacks any basis for drawing distinctions between different classes of outcome: that is, it is not known which outcomes might occur. All that can be said about the outcome is that it will belong to the

class of outcomes of the experiment. All outcomes necessarily belong to this class and hence the uncertainty is once again zero. Between these two extremes lie those situations in which one is partially familiar with the domain: in these one knows enough to know of more than one class of possible outcome but not enough to predict which will occur. In such circumstances the uncertainty takes a finite positive value, corresponding to the number of bits of information one would be given if told the outcome of the experiment.

It is always possible to view the selection of an experience as an experiment whose outcome has an associated prior uncertainty. Hence we are now in a position to produce a simple strategy that models curiosity: always perform the experiment whose outcome has the highest uncertainty, which is equivalent to selecting the experience that will yield the highest information content. Note that it is not necessary for the system to know all possible outcomes before computing the uncertainty because what is required is the *uncertainty of the system's current theory* rather than some absolute notion of the intrinsic uncertainty of the world.

2.3. A control structure for exploratory learning

We are now in a position to propose a simple control structure for exploratory learning, which is shown in figure 1. Step 1 uses the uncertainty heuristic to choose a problem to work on. Step 2 generates experiences relevant to this problem. It is possible that not all the experiences generated in this way will be informative: step 3 is therefore included to filter these experiences (see Markovitch & Scott, 1989 1992) for discussion of experience filtering in learning). Finally, step 4 invokes the theory generator to modify the current domain theory.

This procedure continuously attempts to reduce the uncertainty of the theory. In simple worlds this may result in a steadily declining uncertainty: in more complex worlds the discovery of hitherto unexplored regions of the domain may lead to transient increases in uncertainty that will then be eliminated as exploration extends into the new territory. Upon successful termination, all uncertainty will have been eliminated from the theory which should therefore provide a good model of the domain.

Figure 1. A control structure for exploratory learning.

3. DIDO: A curiosity-driven learning system

We now proceed to describe DIDO, an exploratory learning program we have developed that uses uncertainty to guide its choice of problems and experiences. DIDO's uncertainty-driven problem-choice procedure could be used in conjunction with a wide variety of theory generators, so, since the present article is concerned with methods of choosing problems and searching experience space, we shall not present a detailed account of the methods employed to find better theories. (See Scott & Markovitch, 1991, for a full account.)

3.1. DIDO's task

DIDO interacts with a user-supplied domain composed of a set of discrete objects, each of which has a number of attributes that may take different values. In order for the system to interact with the domain, the user must also supply an appropriate interface comprising a set of perceptual predicates and a set of motor operations:

Perceptual Predicates: Binary predicates. The first argument is an object in the domain; the second argument is a value taken by a specific attribute of that object.

Motor Operations: Unary functions. The argument is an object in the domain. When a motor operation is applied to an object, DIDO receives a list of all the perceptual facts that have changed.

Initially the program has no knowledge either of the kinds of objects that occur in the problem domain, or of the effect upon them of the various operations that comprise the motor system. DIDO's fundamental task is to reach a state in which the outcome of applying any motor operation to any object can be successfully predicted in any situation.

All our experiments to date have been carried out in relatively simple domains characterized by the following restrictions:

Perceptual Predicates: Perceptual predicates have second arguments whose values are nominal constants, because this is the only type of predicate that the theory generator can use to construct class definitions. Thus DIDO "sees" only nominal property values and simple binary relations between specified individual objects. An extension to handle subranges of numerical attributes would be straightforward: more sophisticated extensions would require substantial additions to the theory generator.

Motor Operations: Only simple discrete separable actions whose effects are immediately apparent can be employed, because the theory generator regards all changes as consequences of the immediately preceding motor operation. In the domains we have used, the actions have only affected a single attribute of the object to which they are applied, but this is not an inherent limitation of the program.

Table 1. Perceptual predicates and motor operations in the Adventure domain.

Perceptual Predicate	Possible Values
Type (x, y)	Dwarf, Magician, Pirate
Size (x, y)	Small, Medium, Large
Vitality (x, y)	Alive, Dead
Coat Color (x, y)	Red, Blue, Green, Yellow, Black, White, Gold
Shoes (x, y)	Leather, Boots, Slippers, Sandals, None
Motor Operation	Consequences
Hit With Axe (x)	Kills x if x is a living thing wearing slippers.
Wave Wand (x)	Kills x if x is a living thing wearing boots; Otherwise turns x into a dwarf if x is a magician.
Rub Lamp (x)	Turns x into a dwarf if x is a pirate; Turns x into a pirate if x is a magician; Turns x into a magician if x is a dwarf.

Stability:

All the domains used have been stable in that there are no spontaneous changes and no other agents effecting changes. The system could run unmodified in domains without this restriction but would take longer to discover the effect of its own operations.

Small:

In all the domains used, the number of perceptually distinguishable objects was between a few hundred and a few thousand, while the maximum number of motor operations employed was five. These restrictions were adopted to limit run times. Since each motor operation is learned independently, run time rises linearly with number of operations. Experimental evidence (Scott & Markovitch, 1991) suggests that run times also increase linearly with the number of perceptual predicates.

Table 1 shows an example of a small problem domain that DIDO might encounter if attempting to play a simplified version of the computer game Adventure. The upper portion of the table shows the perceptual predicates available and the values that the second argument may take. The lower table shows the repertoire of motor operations and their consequences. DIDO's task is to construct a theory that will correctly predict which of the 20 possible outcomes will occur whenever any of the three motor operations is applied to any of the 630 distinguishable types of object.

3.2. DIDO's representation scheme

The domain theories that DIDO constructs take the form of an inheritance network of classes whose members are the objects in the domain (see figure 2). Associated with each class is a structure with the following four components:

Figure 2. Final representation produced learning Adventure world (TRUE denotes an intension term that is satisfied by any object).

- Class Intension Term** A necessary condition for class membership that takes the form $P(x,c)$ where P is a perceptual predicate, x is a variable, and c is a nominal class constant. If some object i is a member of the class, then $P(i, c)$ is satisfied.
- Superclass Set** A list of all the direct superclasses of the class.
- Subclass Set** A list of all the direct subclasses of the class.
- Practical Condition Set** A set of practical conditionals (see below).

The complete intension (that is, the set of necessary and sufficient conditions for membership) of a class is the conjunction of the class intension term and all the intension terms of its superclasses. In some circumstances, such as when a class is simply the intersection of its superclasses, no additional class intension term is needed. In such cases it is replaced by the Boolean constant TRUE, which is always satisfied.

3.2.1. Practical conditionals

Practical conditionals serve to represent DIDO's current beliefs about the outcome of applying a specific motor operation to members of a particular class. Each practical conditional has three components:

- Antecedent:* A motor operation $M(x)$. Practical conditionals with the same operation as antecedent may occur in different classes, but a given class will have at most one practical conditional for each operation.
- Consequents:* A list of mutually exclusive outcomes that have been observed when the operation M has been applied to members of the class. Each outcome comprises a specification of a change in a perceptual attribute that has been observed to occur immediately after the operation was applied, together with an estimate of the probability that this outcome will occur. (These probability estimates always sum to 1.) The outcome in which no change is observed is denoted *Null*.
- Uncertainty:* A measure of the uncertainty regarding the outcome that would be obtained if the operation M were applied to a member of the class. It is computed by applying the Shannon information function to the probability estimates of the observed outcomes.

This representation scheme enables DIDO to build domain theories containing alternative hypotheses regarding the outcome of applying an operation to an object. The following example, taken from a run to be described later in this article, shows a practical conditional with three alternative outcomes (x denotes the object to which the operation is applied):

- Antecedent:* Wave Wand(x)
- Consequents:* Type(x , Magician) changed to Type(x , Dwarf): Probability 0.35
 Vitality(x , Alive) changed to Vitality(x , Dead): Probability 0.25
 Null: Probability 0.4
- Uncertainty:* 1.56

This represents the program's past experience of applying the operation Wave Wand to objects that are members of the class whose practical conditional set contains this particular practical conditional: 35% of the time the effect was to transform the object from a magician into a dwarf, 25% of the time the object was killed, while the remaining 40% of the time no changes occurred.

The probability estimates are obtained by applying the following difference equation (known to control engineers as a first-order lag) whenever the operation is applied to a member of the class:

$$P_{t+1} = P_t \times \left(1 - \frac{1}{\tau} \right) + \frac{X_t}{\tau} \quad (2)$$

where τ is a time constant (10 operation applications in the runs reported later), and X_t is 1 if the outcome occurred and 0 otherwise. Provided the probabilities sum to one initially, they will also sum to one after this updating operation, since exactly one outcome occurs. Outcomes whose probability estimates fall below a threshold value δ (0.05 in the runs reported later) are removed and the remaining probability estimates normalized so that they sum to unity. The actual value chosen for τ has a quantitative rather than a qualitative effect on the program's behavior: larger values lead to slower learning but lower sensitivity

to chance variations in the distribution of experiences. The value of δ determines the system's sensitivity to low-frequency events.

A typical practical conditional will contain several outcomes in its consequents, each with a different probability. Excluding the null outcome, the one with the highest probability estimate is called the *principal outcome*.² Thus, in the example above, the principal outcome is the transformation "Type(x, Magician) changed to Type(x, Dwarf)." During learning, the system will attempt to find a subclass in which the principal outcome is the only outcome: this is why the null outcome is excluded, since there seems little value in discovering ways of making nothing happen. When this situation arises, in which a practical conditional has only one outcome whose probability estimate must be 1.0, the practical conditional is said to have *fixated*, since its uncertainty will be zero and hence no further attempt will be made to modify this part of the theory.

3.2.2. Inheritance and representational power

DIDO predicts the outcome of applying a given operation to a particular object in three steps. First, the most specific class that includes the object is found. Second, a practical conditional whose antecedent is the operation must be obtained: if such a practical conditional is in the class's practical conditional set, it will be used; otherwise a practical conditional will be inherited from the most specific superclass whose practical conditional set does include such a practical conditional. The manner in which the class hierarchy is developed guarantees that such inheritance will always be possible: thus the domain theory is always complete (though not necessarily correct). Once a practical conditional has been obtained, its most probable outcome, which will always be either the principal outcome or the null outcome, will be chosen as the predicted outcome.

This representation scheme for domain theories is able to represent any Boolean concepts. Conjunctions are represented by specializations, while disjunctions take the form of alternative subclasses. The procedure of seeking the most specific class, in order to make a prediction, implies that a class is only used to make predictions about those of its members that are *not* members of its subclasses, and thus enables the system to represent *negated* concepts implicitly. An example of a theory for the domain defined in table 2 is shown in figure 2. This is in fact the final version generated in which all practical conditionals have fixated.

Table 2. Learning conjunctive concepts in presence of noise. Final exam scores and uncertainties are means of 20 trials terminated after 1000 examples if not already fixated.

Noise level	Final Exam score	Final uncertainty	Inherent domain uncertainty
0%	100%	0.00	0.00
5%	100%	0.00	0.29
10%	100%	0.03	0.47
15%	100%	0.42	0.61
20%	96%	0.69	0.72
25%	93%	0.82	0.81
30%	89%	0.89	0.88

3.3. *DIDO's theory generator*

We now outline the method DIDO uses to generate improved theories in sufficient detail to enable the reader to understand the operation of the experience selector. (For full details, see Scott & Markovitch, 1991.) Two basic categories of theory modification occur as learning proceeds: changes to practical conditionals and changes to the class hierarchy. The changes to practical conditionals take the form of changes in the probability estimates as a result of experience, corresponding alterations to the uncertainty, addition of new outcomes that were not already represented, and removal of outcomes whose probability estimates have decayed to very low levels. The class structure can be modified in several different ways. Of these the most important are conjectural specialization and retraction.

3.3.1. *Conjectural specialization*

DIDO's experience generator performs batches of experiments in which the same operation, *O*, is applied to a number of objects that are all members of the same class, *C*. If the results of these experiments include occurrences of both the principal outcome, *P*, and at least one other outcome, *conjectural specialization* is invoked. Its purpose is to discover a class for which *P* will be the only outcome. The perceptual predicates satisfied by those objects that gave rise to *P* are compared with the perceptual predicates of those that did not. The perceptual predicate that best discriminates the objects giving rise to *P* is made the intension term of a subclass of *C*, and the practical conditional for *O* is copied from *C* into this subclass. Note that if the subclass does indeed contain all the members of *C* that give rise to *P*, *P* will cease to be the principal outcome of *C* when further experiments are performed.

3.3.2. *Retraction*

Because the subclasses produced by conjectural specialization are conjectured on the basis of a limited amount of evidence, it is possible that they may not in fact include a greater proportion of objects that give rise to the outcome *P* than does the parent class *C*. They may therefore be retracted in the following way. A practical conditional within a class is deemed useless if it does not make predictions that either differ from or have a higher degree of probability than those that would be made by inheriting the corresponding practical conditional from a superclass. Useless practical conditionals are removed from the class that holds them. Classes that contain no practical conditionals and have no subclasses are also deemed useless, and they too are removed.

3.4. *Problem choice and experience selection in DIDO*

Having defined the interface with problem domains, the knowledge representation scheme employed, and the method used to improve the current theory, we can now describe how

Figure 3. DIDO's control structure.

DIDO performs exploratory learning. The basic control structure is given in figure 3. As can be seen, this follows the basic pattern for uncertainty-driven exploratory learning introduced in figure 1.

The outer control loop ensures that DIDO attempts to *minimize the maximum uncertainty* of the theory. Thus learning continues until a theory is found in which all practical conditionals have fixated because their uncertainties have dropped to zero. Step 1 identifies a region of high uncertainty within the current theory, thus defining the problem upon which DIDO should work. The inner loop that constitutes step 2 obtains a set of experiences relating to this region of uncertainty. The random selection process is done without replacement for a given iteration of the outer loop, but all instances are replaced for subsequent iterations. The termination condition for the inner loop depends upon the results of the experiments, thus introducing an empirical component into the experience selection process. The object of attempting to find a mixture of principal and nonprincipal outcomes is to enable the theory generator to compare the two groups in attempting to form a conjectural specialization; thus the procedure also takes account of the method used to search theory space.

DIDO begins learning with a single class, called THINGS, which includes every object that could appear in the domain. A practical conditional for each operation in the motor repertoire is placed in this class. In order to introduce some initial uncertainty, these all take the form:

Antecedent: The operation name (e.g., Kick(x))
Consequents: Something Happened: Probability 0.5
 Null: Probability 0.5
Uncertainty: 1.00

The principal outcome “Something Happened” is a dummy outcome that never actually occurs, and hence will be rapidly discarded and replaced by an observed outcome as learning proceeds.

4. Experimental results

In this section we describe a number of experiments we have performed to investigate the effectiveness of the uncertainty heuristic for choosing problems and selecting experiences. In accordance with our goal of examining the role of experience selection, we shall not discuss those aspects of performance that pertain to the method of generating theories. (For details of the latter and other experimental results, see Scott & Markovitch, 1991).

4.1. Adventure world domain

Our first example involves the domain introduced in the previous section (table 1). The final state, in which all practical conditionals have fixated, is shown in figure 2. This is a complete and correct theory of the domain, and no solution involving fewer classes is possible. In order to see how this result was achieved, we must consider the program’s behavior during the course of learning. Figure 4 shows the results of measuring three parameters as DIDO learned this domain. In each case, the ordinate is the number of operations applied to objects. Figure 4a shows DIDO’s performance on what we term *exams*. During an exam, all of the program’s learning features are turned off. The program is then

Figure 4. (a) Exam scores, (b) mean uncertainty, and (c) error rate learning Adventure world.

presented with 100 randomly chosen operation-object pairs and asked to predict the outcome. As noted in section 3.2.2, DIDO makes predictions by finding the most specific class containing the object and then selecting the most probable outcome from the practical conditional the class holds or inherits for the operation in question. The exam score is the number of correct predictions. In this way an accurate assessment of learning can be obtained without the test itself affecting the learning.

The most interesting feature of the exam score graph is that it reaches a plateau of 100% success a long time before the program terminates. What happens is that the program reaches a state that is extremely close to a complete and correct theory fairly quickly. However, this theory has two limitations not apparent from the exam scores. First, at this stage the uncertainties associated with the various practical conditionals are not zero, so further learning is necessary to eliminate the remaining uncertainty. Second, although the theory is very close to perfect, it may make erroneous predictions in a small fraction of cases: since an exam is made up of only 100 "questions," it does not have sufficient resolution to detect errors in theories that are more than 99% but less than 100% correct.

The two remaining graphs in figure 4 reveal what was happening beyond the point when 100% exam scores were achieved. The elimination of remaining uncertainty is not apparent from the exam score graph, but it is revealed in figure 4b, which shows the average level of uncertainty throughout learning. It can be seen that, after an early rise as DIDO begins interacting with the domain, this declines at a fairly steady rate throughout the run. When it reaches zero all uncertainty has been eliminated and the program halts. It can thus be seen that the question of how long DIDO takes to learn a domain is inherently ambiguous. If unsuccessful prediction is used as the criterion of learning, then the Adventure domain is learned after 125 operations. On the other hand, if confidence that the predictions are correct is used as the criterion, then about three times as many operations are needed. This distinction between being right and knowing that one is right is an important distinction that has received little attention in machine learning, although it is a dominant issue in statistics.

The third graph, figure 4c, reveals another important characteristic of uncertainty-driven learning. This graph shows the error rate, that is, the fraction of incorrect predictions that DIDO makes on the objects *chosen for experiment by the experience selector*. This is quite distinct from the exam score that concerns objects chosen randomly from the whole domain. It is notable that during that portion of the learning when the program is achieving a 100% score in exams, the error rate regularly reaches 20% or higher. What this means is that, during the episodes characterized by peaks on the error rate graph, the uncertainty heuristic is focusing DIDO's learning resources on that tiny portion of the domain that is not yet correctly represented. Such behavior is highly desirable and shows that the uncertainty heuristic is indeed working by seeking out informative examples. An error rate close to zero would indicate that the program had little to learn from the experiments performed. Had the program been choosing examples at random, almost all of them (> 99%) would have served to confirm what the program already knew, and hence it would have taken an extremely long time to reach a complete and correct theory.

Our further characteristic of DIDO's behavior is not apparent from either the final state or the graphs indicating progress, but is an important consequence of the use of the uncertainty heuristic. In the Adventure domain, the program must discover the effects of three

operations. During a run, a few rounds of experiments are performed using one of them, and then another series is performed using another. In this manner DIDO's attention is switched between the three operations throughout the run. The reason is that as the program explores one operation to the point where progress has been made in reducing uncertainty, this operation will no longer be associated with the most uncertain parts of the theory: hence attention is switched to one of the other operations, and will only return when comparable progress has been made with them. This is a highly desirable mode of behavior in that it distributes the available learning resources evenly across the repertoire of motor operations.

4.2. Inconsistent domains: Noise and non-determinism

The Adventure domain appears consistent to DIDO: that is, the same operation will always lead to the same observable effect if applied to identical objects. Since DIDO's goal is to construct a consistent theory of its domain, it is interesting to consider how the program will behave in worlds that appear to behave inconsistently. Such inconsistency could arise either through the presence of *noise* (random disturbance of the processes of action and observation), or because the domain itself is *non-deterministic*.

Figure 5 shows the results obtained when DIDO was required to explore domains with various levels of noise present. In these domains there was only one operation which had only two possible outcomes: thus the task is effectively one of classification. Objects had five attributes, each of which had three alternative values. In the runs reported, the program had to discover a conjunction of two properties to make successful classifications. Various authors have defined noise level in different ways (e.g., Quinlan, 1986; Langley, 1987). In our experiment it is defined as follows:

Let o_1 and o_2 be the outcomes, and N the noise level expressed as a percentage.

If the outcome of applying a particular operation to a specific object in a noise-free domain would always be o_1 , then the outcome will be o_1 for $(1 - N)\%$ and o_2 for $N\%$ of operation applications.

If the outcome of applying a particular operation to a specific object in a noise-free domain would always be o_2 , then the outcome will be o_2 for $(1 - N)\%$ and o_1 for $N\%$ of operation applications.

This is equivalent to combining positive and negative noise following Langley's (1987) definition.

Since this experiment concerns a domain that appears inconsistent because of the presence of noise, we wished to investigate how successful DIDO was in discovering the underlying deterministic domain. The noise level was therefore set to 0% during all exams. The results are shown in figure 5a. For levels of noise up to 15%, DIDO eventually attained 100% exam score, indicating that the program had acquired a theory that correctly modeled the underlying domain. The runs with 0% and 5% noise fixated well before the 1000 operations had been completed: at higher noise levels DIDO was still attempting to find a better theory at the end of the run. Otherwise, the only effect of the noise appears to be to slow down the rate of learning.

Figure 5. (a) Exam scores and (b) uncertainty in non-deterministic domains.

For levels of noise above 15%, DIDO never reaches a 100% exam score: the program's behavior at these higher levels can be interpreted as an attempt to construct a non-deterministic theory of the domain. While we chose to describe the random changes made to outcomes as noise, they could equally well have been regarded as a consequence of non-deterministic rules in the domain. Under this latter interpretation, an *inherent domain uncertainty* can be associated with each domain. This is defined as the uncertainty that an observer who knew the non-deterministic rule would have about the outcome of an operation. Figure 5b shows the average uncertainty during the same runs. For low noise levels the uncertainty fell to zero, or close to it, but for higher levels a substantial amount of uncertainty remained at the end of 1000 operations. These final uncertainty values are given in table 2, together with calculated values for the inherent domain uncertainty. It is clear that at high levels of noise, the final state of the system is one in which the uncertainty of the theory corresponds to the inherent uncertainty of the domain.

Thus DIDO has two modes of behavior when confronted with apparently inconsistent domains. Low levels of inconsistency will be treated as being due to noise and the program will construct a theory for the underlying deterministic domain. At higher levels of inconsistency, the program is unable to develop such a deterministic theory but will build a theory

whose non-determinism directly reflects that of the domain. The transition between these two modes occurs when the noise level is such that it is less likely that a correct practical conditional will fixate than it is that further classes will be added in an attempt to produce a theory that accounts for the effects of noise.

The noise level at which this occurs is largely determined by the system parameters τ , the time constant for estimating probabilities, and δ , the threshold below which probability estimates are regarded as negligible (see section 3.2.1). A practical conditional will fixate if the probability estimates of all but one outcome fall below δ . Consider the situation in which a practical conditional is very close to fixating on the hypothesis that a specific motor operation always leads to the outcome o_d when applied to members of a particular class. Suppose further that this hypothesis correctly models the underlying deterministic domain but that, because of noise, the outcome o_d occurs with probability p and some other outcome o_n occurs with probability $(1 - p)$. A single occurrence of outcome o_n will prevent the immediate fixation of the practical conditional. By equation (2), the probability estimate for outcome o_n will have fallen below δ after a further K consecutive occurrences of outcome o_d , where K satisfies

$$\frac{1}{\tau} \left(1 - \frac{1}{\tau} \right)^K < \delta. \quad (3)$$

Hence, with the parameter values used in these experiments ($\tau = 10$, $\delta = 0.05$), $K \geq 7$. Thus the probability that a correct practical conditional will fixate is the probability of getting 7 successive occurrences of outcome o_d , which is p^7 . Hence, if the noise level is 10% (corresponding to $p = 0.9$) the probability that a correct practical conditional will fixate is 0.48; if the noise level is raised to 20%, the fixation probability will drop to 0.21. Thus for noise levels above 10%, DIDO is more likely to extend the domain theory, in an attempt to account for the effects of noise, than to fixate with a theory corresponding to the underlying deterministic domain.

When the noise level is high, DIDO's asymptotic state is one in which it is vainly attempting to find deterministic rules that predict the random behavior of the domain. We have investigated the possibility of enabling the program to recognize such a situation and hence give up its fruitless search. We provided DIDO with a facility for monitoring the change in uncertainty and abandoning the search if no significant reduction had occurred for some time. Unfortunately, this was not successful because the uncertainty fluctuates considerably in the asymptotic state as practical conditionals are repeatedly conjectured and retracted. Thus transient drops in the uncertainty were regarded as evidence of continued progress, even though they were shortly offset by rises. Attempts to remedy this difficulty by considering the average change in uncertainty over a time interval created the opposite problem: the program would detect no apparent progress when progress was still possible. Hence, while the program is able to build a non-deterministic theory of a non-deterministic domain, we were not able to provide a way of recognizing that it has done so. Consequently, for the present, DIDO, like Einstein in the real universe, stubbornly refuses to accept that God plays dice in its simple domains.

5. Discussion

5.1. Conclusions

These results demonstrate that uncertainty works effectively as a problem choice and experience selection system in DIDO. Could it also be employed with comparable success in other systems? Or more specifically, what characterizes those learning systems in which the uncertainty heuristic could be used successfully? The fundamental requirement is that the representation scheme for theories must include some means of holding alternative hypotheses regarding some facet of the domain. Many representation schemes have this characteristic explicitly—for example, the retention of alternative hypotheses in Mitchell's (1982) version space algorithm, and the storage of sets of examples in instance-based representations (Aha, Kibler, & Albert, 1991). However, in principle, the heuristic could be employed with any theory representation scheme that has at least one of the following two characteristics:

- Inconsistency:* During learning, domain theories may contain mutually contradictory alternative hypotheses.
- Non-determinism:* Domain theories may make non-deterministic predictions, associating some measure of likelihood with each alternative prediction.

DIDO's representation scheme stores estimates of the probabilities for the various outcomes, and this facility allows the use of the Shannon function to define uncertainty. Even if such information is not available within a representation scheme, it should be possible to use the Shannon function and obtain most of the benefit of the uncertainty heuristic by treating all outcomes as equally likely. However, it is clearly preferable to have the information about outcome probabilities within the representation. Furthermore, any representation that does store such information will have the additional benefit of allowing the system to attach confidence levels to any decision it makes.

Our principal conclusion is therefore that the uncertainty heuristic provides an effective means of choosing problems for any exploratory learning system that stores alternative predictions in an explicit form, and is particularly suited to those that also store information about the likelihoods of those alternatives.

5.2. Limitations and further work

The current version of DIDO operates in comparatively simple domains. Increasing the size of the domain may lead to problems if the result is that some events occur so rarely that they are overlooked. DIDO's sensitivity to low-frequency events is determined by τ , the time constant for estimating probabilities, and δ , the threshold for discarding low-probability outcomes. Some mechanism for varying these parameters as learning progresses may be needed to handle significantly larger domains.

The restriction of the perceptual system to two-argument predicates with constant second arguments is imposed by DIDO's current theory generator. It is clearly desirable that

DIDO should be able to handle a much richer set of relations. For example, given a blocks-world domain, the current theory generator may discover a rule stating that any block placed on one specific pyramid would fall off, but it would be unable to discover a more general rule stating that any block will fall off any pyramid. An extension allowing DIDO to consider such relations, which involve arbitrary members of classes rather than individuals, would greatly enlarge the theory space to be searched, since any individual will be a member of many classes.

The restriction of motor operations to simple discrete separable actions is more fundamental and is shared with the large body of artificial intelligence research on planning. The requirement that the outcome be immediately apparent could only be removed if the theory generator were provided with some memory of its actions, together with some method of learning associations between temporally separated events. The discovery of temporal correlations is a problem that has been neglected by machine learning researchers. This is surprising, given the important role such correlations play in discovering causal relationships: it is clearly a topic that merits further research.

The restriction of the system to domains in which there are no spontaneous changes is not very serious if the rate of spontaneous change is low: such changes will only coincide with specific actions rarely and hence will be ignored as noise. However, if the spontaneous change rate is high, DIDO is likely to develop erroneous hypotheses through regarding changes that were spontaneous as consequences of the system's actions. To avoid this difficulty, an extension would be needed that enabled DIDO to maintain records of how often changes occurred in the absence of any action.

DIDO is also restricted to domains in which there are no other agents changing the world. In domains without this restriction, the other agents would be by far the most interesting objects: a major portion of the learning task would be that of discovering what the other agents are, what they can do, and in what circumstances they are likely to do it. This is a topic rich with research possibilities.

Acknowledgments

The work reported in this article was supported by NSF Grant #MCS-8203956. Thanks are due to Xerox Corporation, University of Michigan, and the Center for Machine Intelligence, Ann Arbor, for use of computing facilities, and to Bob Vogt and John Wiersba, who assisted in the implementation of an early version of DIDO.

Notes

1. In earlier publications we used the term *representation generator* for this component of a learning system.
2. The principal outcome is called the *expected outcome* in Scott and Markovitch (1991).

References

- Aha, D.W., Kibler, D., & Albert, M.K. (1991). Instance-based learning algorithms. *Machine Learning*, 6, 37–66.
- Carbonell, J.G., & Gil, Y. (1987). Learning by experimentation. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, California (pp. 256–265), San Mateo, CA: Morgan Kaufmann.
- Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*, pp. 99–161. Cambridge, MA: MIT Press.
- Langley, P., Simon, H.A., Bradshaw, G.L., & Zytkow, J.M. (1987). *Scientific discovery: Computational explorations of the creative process*. Cambridge, MA: MIT Press.
- Langley, P., & Zytkow, J.M. (1989). Data-driven approaches to empirical discovery. *Artificial Intelligence*, 40(1–3).
- Markovitch, S., & Scott, P.D. (1989). Information filters and their implementation in the SYLLOG system. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 404–407), Ithaca, New York. Los Altos, CA: Morgan Kaufmann.
- Markovitch, S., & Scott, P.D. (1993). Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10, 113–151.
- Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence*, 18, 203–226.
- Quinlan, J.R. (1986). The effect of noise on concept learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.
- Rajamoney, S.A. (1989). Exemplar-based theory rejection: An approach to the experience consistency problem. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 284–289), Ithaca, New York. Los Altos, CA: Morgan Kaufmann.
- Rajamoney, S., DeJong, G., & Flatings, B. (1985). Towards a model of conceptual knowledge acquisition through directed experimentation. *Proceedings of the Ninth International Conference on Artificial Intelligence* (pp. 688–690). Los Angeles: Morgan Kaufmann.
- Ruff, R.A., & Dieterich, T.G. (1989). What good are experiments? *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 109–112), Ithaca, New York. Los Altos, CA: Morgan Kaufmann.
- Scott, P.D., & Markovitch, S. (1989a). Uncertainty based selection of learning experiences. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 358–361), Ithaca, New York. Los Altos, CA: Morgan Kaufmann.
- Scott, P.D., & Markovitch, S. (1989b). Learning novel domains through curiosity and conjecture. *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 669–674). Detroit: Morgan Kaufmann.
- Scott, P.D., & Markovitch, S. (1991). Representation generation in an exploratory learning system. In D. Fisher & M. Pizzani (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.
- Shalin, V.L., Wisniewski, E.J., Levi, K.R., & Scott, P.D. (1987). A formal analysis of machine learning systems for knowledge acquisition. *Proceedings of the Second AAAI Knowledge Acquisition For Knowledge Based Systems Workshop*. Banff, Canada. An extended version appears in B.R. Gaines & J.H. Boose (Eds.), *Machine learning and uncertain reasoning: Knowledge based systems* (Vol. 3). San Diego, CA: Academic Press, 1990.
- Shannon, C.E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana, IL: University of Illinois Press.
- Shen, W.M., & Simon, H.A. (1989). Rule creation and rule learning through environmental exploration. *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 675–680). Detroit: Morgan Kaufmann.
- Simon, H.A., & Lea, G. (1974). Problem solving and rule induction: A unified view. In L.W. Gregg (Ed.), *Knowledge and cognition*, pp. 105–127. New Jersey: Lawrence Erlbaum.
- Winston, P.H. (1975). Learning structural descriptions from examples. In P.H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.
- Zytkow, J.M. (1987). Combining many searches in the FAHRENHEIT discovery system. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 281–287), Irvine, California. Los Altos, CA: Morgan Kaufmann.
- Zytkow, J.M., & Zhu, J. (1991). Automated empirical discovery in a numerical space. *Proceedings of the Third Annual Chinese Machine Learning Workshop '91*.

Received December 5, 1990

Final Manuscript May 26, 1992