PAGODA: A Model for Autonomous Learning in Probabilistic Domains

Marie Ellen desJardins

A dissertation submitted to the Department of Computer Science of the University of California, Berkeley, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

PAGODA: A Model for Autonomous Learning in Probabilistic Domains

by

Marie Ellen desJardins

Abstract

Machine learning approaches have traditionally made strong simplifying assumptions: that a benevolent teacher is available to present and classify instances of a single concept to be learned; that no noise or uncertainty is present in the environment; that a complete and correct domain theory is available; or that a useful language is provided by the designer. Additionally, much existing machine learning research has been done in a piecemeal fashion, addressing subproblems without a uniform conceptual approach to designing intelligent systems. The resulting learning techniques often are only useful for narrowly defined problems, and are so dependent on the underlying assumptions that they do not generalize well—or at all—to complex domains.

PAGODA (Probabilistic Autonomous GOal-Directed Agent), the intelligent agent design presented in this thesis, avoids making any of the above assumptions. It incorporates solutions to the problems of deciding what to learn, selecting a learning bias, and inductive learning under uncertainty, in an integrated system, based on the principles of probabilistic representation of knowledge, Bayesian evaluation techniques, and limited rationality as a normative behavioral goal. PAGODA has been implemented and tested in a simulated robot domain, RALPH (Rational Agent with Limited Performance Hardware).

Goal-Directed Learning (GDL) allows the agent to decide what to learn, enabling autonomous learning in complex domains. The value of being able to predict various features of the environment is computed using the principles of decision theory. The agent uses the features with highest values as *learning goals* for building predictive theories.

Probabilistic Bias Evaluation (PBE) determines the learning bias for each learning goal using probabilistic domain knowledge, an expected learning curve, and a time-preference function to find the expected discounted future accuracy for proposed biases; the best of these biases is used for learning.

Theories are represented as Uniquely Predictive Theories (UPTs), which consist of restricted sets of conditional probabilities. Probability Combination using Independence (PCI), a probabilistic inference method which relies on minimal independence assumptions, is applied to the theories to make probabilistic predictions for planning and evaluation. A Bayesian evaluation method is used to determine the best theory to explain the observed data.

Chapter 1 of the thesis defines the problem of building autonomous rational agents, and motivates PAGODA as a solution to this problem. Chapter 2 surveys past approaches to probabilistic learning. Chapter 3 describes PAGODA's performance element, including the RALPH world and PAGODA's probabilistic representation for theories (UPTs), inference method (PCI), and planning mechanism. Chapters 4, 5, and 6 describe Goal-Directed Learning, Probabilistic Bias Evaluation, and probabilistic learning, respectively. The implementation of PAGODA in the RALPH domain and results of empirical tests are described in Chapter 7. Related work in a number of fields is discussed in Chapter 8. Finally, Chapter 9 presents conclusions and outlines open problems for future research.

To John

Contents

1 Introduction 1
- 1.1 The Problem of Autonomous Learning . . . 2
- 1.2 Central Issues . . . 5
- 1.3 PAGODA: An Autonomous Agent Model . . . 8

2 Survey of Probabilistic Learning 14
- 2.1 Probability Theory . . . 14
- 2.2 Representing Probabilistic Knowledge . . . 16
  - 2.2.1 Logic and Probability . . . 17
  - 2.2.2 Belief Networks . . . 19
- 2.3 Probabilistic Learning Techniques . . . 21
  - 2.3.1 Maximum Entropy . . . 21
  - 2.3.2 Minimum Length Encoding . . . 22
  - 2.3.3 Probabilistic Machine Learning . . . 23
  - 2.3.4 Planning to Learn . . . 25

3 Performance Element 28
- 3.1 RALPH: An Autonomous Agent Testbed . . . 28
- 3.2 Representing Probabilistic Knowledge . . . 34
  - 3.2.1 Conditional Probability . . . 35
  - 3.2.2 Conditional Distributions . . . 36
  - 3.2.3 Predictive Theories . . . 37
  - 3.2.4 Uniquely Predictive Theories . . . 38
- 3.3 Theory Notation . . . 42
- 3.4 Probabilistic Inference . . . 43
  - 3.4.1 An Example of PCI . . . 45
- 3.5 Planning . . . 47

4 Goal-Directed Learning 50
- 4.1 Utility of Learning Goals . . . 52
- 4.2 Utility of Plans . . . 55

4.3 An Example in the RALPH World 57

5 Selecting a Learning Bias **61**
5.1 Background 63
5.1.1 Bias in Machine Learning 64
5.1.2 Declarative Bias 65
5.1.3 Minimum Length Encoding 66
5.2 Probabilistic Evaluation of Bias 67
5.3 Probabilistic Background Knowledge 69
5.4 Expected Accuracy 71
5.5 Learning Curves 73
5.6 Time Preference Functions 75
5.7 Expected Value of Biases 76
5.8 Results 77
5.8.1 Learning Procedure 77
5.8.2 Analysis of Results 78

6 Probabilistic Learning **82**
6.1 Theory Evaluation 84
6.1.1 Evaluating Probabilistic Theories 85
6.1.2 Bayesian Probability 86
6.2 Prior Probability 87
6.2.1 Uniform Distribution on Theories 90
6.2.2 Rule-level Classification of Theories 91
6.2.3 Feature-level Classification of Theories 92
6.2.4 Huffman Encoding of Terms 93
6.2.5 Expected Effects 94
6.3 Likelihood of Evidence 95

7 Implementation and Results **99**
7.1 PAGODA-RALPH Implementation 99
7.1.1 Overall Behavior 99
7.1.2 Bias Evaluation 100
7.1.3 Hypothesis Generation and Evaluation 101
7.1.4 Goal Generation 103
7.2 Tests and Results 103
7.2.1 Testing Biases and Learning 105
7.2.2 Testing Priors 112
7.2.3 Testing Learning Goals 118
7.3 Conclusions 123

8 Related Work **125**
8.1 Classification of Machine Learning Research 125
8.1.1 Ideal Autonomous System 127
8.1.2 Classification of PAGODA 128
8.2 Inductive Learning 129
8.3 Autonomous Learning 130
8.4 Bias 133
8.4.1 Declarative Bias 134
8.4.2 Shift of Bias 135
8.5 Planning 136
8.5.1 Reactive Strategies 137
8.5.2 Deliberative Planning with Uncertainty 138

9 Future Work and Conclusions **140**
9.1 Goal-Directed Learning 140
9.2 Selecting a Learning Bias 143
9.3 Probabilistic Learning 148
9.4 Probabilistic Planning 151
9.5 Conclusions 153

Bibliography **154**

A Maximum Entropy Proof **165**

B ID* **169**
B.1 Description of ID* 169
B.2 Description of the Test Domain 172

C Generating Internal States **175**
C.1 Dichotomization 177
C.2 An Example in the RALPH World 178

Acknowledgments

I would probably never have begun graduate school, much less finished it, if it had not been for the lifelong support of my parents, Mary and Richard. They inspired me to work hard, to aim high, and to enjoy life.

Lotfi Zadeh encouraged me to come to Berkeley, and provided me with financial support for a year; he also suggested that I should switch research advisors when Stuart Russell joined the faculty. Stuart's fortuitous arrival allowed me to pursue my interest in machine learning. Our many meetings and discussions over the years have helped to shape my understanding of the important problems in the field, and have honed my debating skills greatly.

My ideas and opinions about artificial intelligence in general, and machine learning in particular, were also influenced by discussions and arguments with fellow graduate students. Thanks go to the RUGS (Russell's Unusual Group of Students) research group at Berkeley for listening to talks, reading papers, and providing invaluable feedback and advice: Lonnie Chrisman, Jeff Conroy, Othar Hansson, Tim Huang, Mike Malone, Sonia Marx, Andy Mayer, Gary Ogasawara, Ron Parr, Sudeshna Sarkar, Shlomo Zilberstein, and the late Eric Wefald. I would especially like to thank Ron Musick, who managed to read the entire dissertation—and even to find it interesting!

Alice Agogino and Richard Fateman, the other members of my qualifying exam committee, were among the first people to express an interest in my work. Peter Cheeseman, Doug Fisher, and Pat Langley have all provided helpful comments and advice, and positive feedback, an essential ingredient to any thesis.

Haym Hirsh was the first person to state publicly that my research was interesting; not surprisingly, we immediately became friends. I only wish we had

met earlier; in a few days, our discussions gave me nearly as much enthusiasm for continuing my research as I had built up over the course of the previous six years.

When I was in my hour of darkest need, trying to develop a deeper understanding of probability theory, Dekai Wu was doing the same. Studying together helped us both immensely. Dekai also spent a long afternoon helping me to polish the ideas that eventually became PBE. In our race to take the longest to finish our dissertations, I won by a month.

Ann Almgren, Michael Braverman, Lise Getoor, and Marti Hearst commiserated regularly with me about the shared challenges of life as a graduate student. Penny Rheingans, my closest friend, provided a sympathetic ear whenever I needed one, and always believed I would finish.

It has always been important to me to balance my life between computer science and other activities. Thanks to Jeannette Hung, Andrew Neuschatz, Kathy Post, and many others for dragging me away from the terminal with alarming regularity. Thanks also to all of the people I have sung with over my years at Berkeley, especially Heather Bourne, for enriching and enlivening my life.

Marie Bienkowski and Roberto Desimone gave me a terrific job at SRI International, leading to five of the most exciting (and most difficult, thanks to the unfinished dissertation) months of my life. I look forward to continuing to work with them under less stressful conditions.

The administrative staff of Berkeley's CS department have always been friendly, helpful, and essential for dealing with the morass of bureaucracy. Thanks to Kathryn Crabtree, Teddy Diaz, Liza Gabato, Danny Howard, and Jean Root.

Finally, I will never be able to describe the depth of love, support and caring that my husband, John Park, has given to me. During our nine years together, he has helped me to grow as a scientist, as a teacher, and most of all as a person. Without his patient understanding and unshakeable faith in my abilities, I doubt I would have made it through the tough times. Having him there to share the successes and small triumphs made them more meaningful. I thank him, most of all, for making it all matter.

This work was supported in part by a NASA Graduate Student Researcher fellowship, and by a grant from Lockheed.

Chapter 1

Introduction

The trend in artificial intelligence research has been to decompose the problem of intelligence, focus on a single aspect (e.g., planning or learning) and then to work on a subproblem within that problem area (e.g., building an optimal decision tree or constructing new features for a learning bias). This approach results in specialized systems that do not generalize well to domains or problems other than those they were designed for, and that only work in isolation, without connecting naturally to solutions to the remaining problems.

Additionally, machine learning approaches have often made strong simplifying assumptions; for example, that a benevolent teacher is available to present and classify instances of a single concept to be learned; that no noise or uncertainty is present in the environment; that a complete and correct domain theory exists; or that a useful language is provided by the designer.

This thesis describes PAGODA (Probabilistic Autonomous GOal-Directed Agent), a model for an intelligent agent which avoids the problem of overly specialized focus, and does not rely on the simplifying assumptions mentioned in the previous paragraph. PAGODA consists of a flexible, extensible architecture for an intelligent agent that addresses a number of open problems in machine learning. It incorporates solutions to the problems of selecting learning tasks, choosing a learning bias, classifying observations, and performing inductive learning under uncertainty, in an integrated system.

The guiding principles behind PAGODA include probabilistic representation of knowledge, Bayesian evaluation techniques, and limited rationality as a normative behavioral goal. The key properties of PAGODA are:

- The agent operates autonomously, with minimal intervention from humans, and does not require a teacher to present or classify learning instances, or to provide a representation for learned theories.
- The agent handles uncertainty due to inaccurate sensors, randomness in the environment, and sensory limitations. The learned theories express observed uncertainty explicitly.

Most past machine learning systems have not included either of these properties, and very few have included both. The development of PAGODA highlighted the fact that the interactions that arise when building an agent with both of these properties are complex and not well understood. One of the most important contributions of this thesis is to identify and analyze these interactions.

The domain on which this thesis focuses is an autonomous mobile robot manipulating and being affected by a complex, nondeterministic environment. Currently, PAGODA is implemented in the RALPH (Rational Agent with Limited Performance Hardware) world, an intelligent agent testbed at UC Berkeley. The RALPH world is described in Chapter 3.

The remainder of this chapter is organized as follows: Section 1.1 defines the problem of autonomous machine learning. Section 1.2 describes some of the most important issues that arise when designing an integrated intelligent agent. An overview of PAGODA is given in Section 1.3.

1.1 The Problem of Autonomous Learning

In this section, we define the concepts of embedded limited rational agents and autonomous learning; these two concepts are central to the thesis.

An embedded agent consists of three components: a transducer, which passes information from the environment to the agent as sensory observations and provides

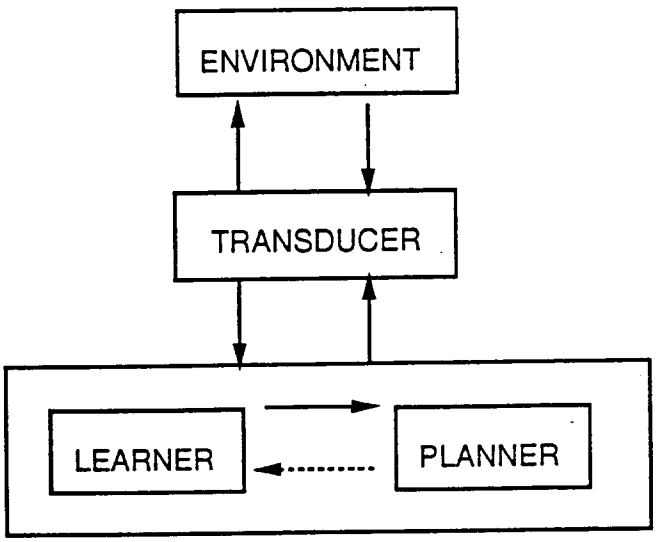

Figure 1.1: Embedded Agent Model

the motor controls that allow the agent to move about in and manipulate the environment, a learning module, and a planner. Embedded agents must interact with their environments in real time, and are continuously being affected by and manipulating the environment. If any other agents (such as a teacher) are present, they are simply viewed as part of the environment. A schematic view of an embedded agent is shown in Figure 1.1.

The sensory input, which may include visual information, sound, infrared, or anything else the detectors can provide, gives the agent a window on the environment. This information may be incomplete (in that it does not represent the entire state of the environment) or inconsistent (because the sensors may not always give identical readings in identical states). The agent's sensors provide a (possibly nondeterministic) mapping from the state of the world to values of the sensory inputs. We will refer to the values of the inputs at any given time as the agent's perceptions, or its perceived world. A perceived world may correspond to many actual world states, which the agent may or may not be able to learn to distinguish.

The agent's actions allow it to move about in limited ways, usually with limited accuracy, so that executing the same action in the same world state will not always result exactly in the same outcome.

The learning module uses the information provided by the sensors and whatever background knowledge it has available (from earlier learning, or from the designer) to build a world model or otherwise provide information that the planner can use to select actions. In our model, the agent initially knows what actions it can execute, but not what effect its actions have on the environment. PAGODA builds an explicit, predictive world model that the planner uses to construct deliberate plans, but in general, the learner may construct any sort of knowledge usable by a planner, such as condition-action rules (reactive strategies) or a neural network. Russell [1989] characterizes the forms of knowledge that an agent may have about the world along a continuum from declarative (e.g., predictive rules) to compiled (e.g., rules specifying the best action to take in a given situation). PAGODA's world model would be classified by this model as purely declarative.

A *rational* agent chooses actions that maximize its expected utility. An embedded rational agent must have strategies for learning and planning that maximize its expected utility in the current environment.

A *limited* rational agent takes into account the cost of time, exhibiting a strategy that maximizes the expected average utility per unit time by balancing time spent deliberating with actually performing external actions. In an embedded agent, since deliberating includes both learning and planning, the utility gained from these activities must be traded off with their time cost.

An *autonomous* agent operates independently of human intervention. Specifically, it does not require inputs (except for its initial state) to tell it what its goals are, how to behave, or what to learn. In this thesis, an autonomous agent will always mean an autonomous, limited, rational, embedded agent.

The problem of *autonomous learning* for a limited rational embedded agent is: given sensory inputs and background knowledge, incrementally learn a model of the world (or whatever representation the planner requires) that allows the planner to maximize expected utility per unit time. Note that this does not mean that the learner must learn a perfect world model, because that might require so much time that the agent doesn't have enough time left to use the model effectively. On the other hand, if the agent allocates all of its time to planning, and none to learning,

it will not choose good actions. The learner must learn a good world model quickly enough so that the planner has time left to react to the world and plan strategies to maximize utility.

1.2 Central Issues

In this section we discuss four specific problems that arise from the definition of autonomous learning given in the previous section. These problems are those of deciding what to learn, selecting learning representations, learning in uncertain domains, and planning under uncertainty.

Deciding What to Learn

In a complex environment, the true world model (i.e., the underlying state-transition function) will be too complicated for an agent with limited resources to learn completely in a reasonable amount of time. Therefore, in order to be useful, the agent will have to focus attention on learning portions of this true world model. A rational agent should allocate its resources so as to maximize its ultimate goal achievement, by focusing its learning attention on whatever aspects of the world are expected to be most useful to learn.

Supervised learning systems such as ID3 [Quinlan, 1986] and CIGOL [Muggleton and Buntine, 1988] require a teacher to select concepts for the system to learn, and to present and classify instances of these concepts. However, for general autonomous agents, it is not realistic to expect an altruistic teacher to be present at all times to guide learning. The case of an AI-based Mars probe is an obvious example, where the large time delay between the system and human controllers will require largely autonomous functioning. Even in more mundane applications such as robot household helpers for the handicapped or robot street sweepers, designers must expect that the system will encounter unforeseen situations which it must learn to deal with. In such cases, being able to decide what to learn will be essential

Of course, a rational agent will use whatever sources of information are available, including the environment, potential teachers, and reference manuals, to

learn and to plan courses of action. All of these sources must be treated by the agent as sensory inputs to be processed, interpreted, and used to make decisions.

Selecting Representations for Learned Theories

An autonomous agent must not only select its own learning tasks (i.e., features of the world to be predicted) but must also decide what properties of the world are relevant in predicting these features. This is part of deciding what *bias* to use for each learning task. Bias, as defined by Mitchell [1980], is

> ...any basis for choosing one generalization over another, other than strict consistency with the observed training instances.

Bias can result from constraining the representation language used to express learned theories, or by expressing a preference within the space of allowed theories. A completely unbiased learner could never generalize: the disjunction of all observed instances would always be considered as good a theory as any other. Additionally, bias is necessary for efficient learning. In a complex domain, there may be a large number of irrelevant features of the environment present in any given learning problem. For example, when an agent wants to learn a rule to predict how frequently parking meters are checked, relevant properties of the world may include the day of the week and time of day, weather, and how busy the street is; but the color of its car, how wide the sidewalk is, and what it had for breakfast probably are not useful. Ignoring these irrelevant features saves time without loss of predictive accuracy of learned theories.

Rather than requiring the designer to specify biases for each potential learning task, we believe that it will be necessary for the agent to use whatever domain knowledge it has been given or has learned to select biases as learning tasks arise. This domain knowledge can tell the agent what sorts of bias are appropriate for classes of learning tasks.

Learning in Uncertain Domains

Many real-world environments contain uncertainty, which can arise from randomness in the world, noise in the agent's sensors, sensory limitations of the agent,

and complexity. In order for an agent to function in such an environment, it cannot expect the world to be deterministic. It must have mechanisms for handling noise in its input and, ideally, a representation that allows it to express the uncertainty that is present in the world.

Traditionally, learning has been defined as the problem of finding a theory that is consistent with all observed instances (see, for example, [Carbonell *et al.*, 1983] and [Board and Pitt, 1989]). However, when uncertainty is present in the form of randomness in the environment or noise in the agent's sensors, there may be no consistent theories under a reasonable learning bias (e.g., one which does not allow the disjunction of all observations to be used as the theory). Because of this, traditional learning approaches do not generalize well to domains containing uncertainty.

Another source of apparent uncertainty is complexity. If there are rare exceptions to a general rule, although the agent may be able to determine when these exceptions occur, it may not be worth expending its limited resources in doing so. A limited rational agent will have to decide when it is worth finding and representing these exceptions. This decision will depend on the expected gain in utility of representing the exception versus the expected cost of doing so.

Finally, if the learner does not represent uncertainty (e.g., if it only stores the most likely outcome for each situation), the agent will not be able to determine all of the potential consequences of its actions, and will therefore be incapable of maximizing its utility. Uncertainty must be dealt with explicitly by a limited rational agent, and can only be ignored when the agent decides that that is the rational thing to do.

Planning Under Uncertainty

When an agent's learned world model contains uncertainty, the agent needs a planning mechanism that can decide how to maximize goal satisfaction in the face of this uncertainty. Classical AI planning techniques require deterministic models of the world, and therefore are inapplicable to the domains we are interested in. Fortunately, decision theory provides us with a paradigm for behaving optimally under uncertainty.

Decision theory requires the agent to choose whatever action maximizes its

expected future average utility per unit time. For a limited rational agent, this action-selection must include deciding whether to think more, or whether to choose the best action determined so far. This decision-making process, called metareasoning, can in theory lead to infinite regress (how does the agent decide how to decide, and so forth). Although we do not address these issues here, there is active research being done in this area. (See, for example, [Russell and Wefald, 1991]).

1.3 PAGODA: An Autonomous Agent Model

PAGODA (Probabilistic Autonomous GOal-Directed Agent) is a limited semi-rational embedded agent that exhibits autonomous learning. We say "semi-rational" because PAGODA does not exhibit *optimal* resource-bounded behavior. However, the model does explicitly consider issues of limited rationality in its processes, providing important contributions towards building an optimal agent. PAGODA consists of four major components, and an architecture in which they are applied. The four components—Goal-Directed Learning, Probabilistic Bias Evaluation, probabilistic learning, and probabilistic planning—are described in the following paragraphs.

Goal-Directed Learning

Initially, PAGODA has a trivial "theory" for features in its sensory inputs, in the sense that it can determine their values by examining its sensory inputs. Its learning effort is directed towards being able to *predict* feature values resulting from a proposed sequence of actions, by forming a model of the world that provides a mapping from perceived worlds and actions to resulting perceived worlds. It can then use this model to choose actions that maximize its expected utility. However, given that it has limited resources, it must constrain the scope of its world model to cover only the aspects of the world that are most relevant to its ability to maximize utility.

We have developed an approach called Goal-Directed Learning (GDL), which allows the agent to decide what features of the world are most worth learning about. The agent uses decision theory to compute the information value (i.e., the expected utility) of knowing various features of the world, and uses the features with the

greatest value as its *learning goals*, i.e., as features to predict. The value of a learning goal is the difference between the agent's expected utility when the learning goal can be predicted and its expected utility when it cannot be predicted. GDL is described in detail in Chapter 4.

Evaluating Learning Biases

PAGODA uses probabilistic background knowledge to evaluate potential biases for each learning task by computing how well each bias is expected to perform during future learning. The chosen bias may be changed later if the agent's theories are not predicting the world as accurately as it expected.

Probabilistic Bias Evaluation (PBE) chooses a set of features that is as relevant as possible, without being so large that the complexity of the learning task is excessive. Each potential bias, consisting of a set of features to be used in predicting the learning goal, is assigned a value. This value is determined using a decision-theoretic computation which combines the expected accuracy of predictions over time with a time-preference (or discounting) function that expresses the agent's willingness to trade long-term for short-term performance. The computed value represents the expected discounted accuracy of predictions made by theories learned using the given bias. The bias with the highest value is used for learning. PBE is described in Chapter 5.

Probabilistic Learning

PAGODA represents its learned theories as Uniquely Predictive Theories (UPTs), which consist of sets of conditional probabilities meeting certain constraints. The probabilities specify the distribution of outcomes of PAGODA's learning goals, given a state of the world and an action. Each theory contains probabilities for predicting a different learning goal; the world model is therefore a collection of theories. A probabilistic inference mechanism, PCI (Probability Combination using Independence), allows PAGODA to make predictions about the outcomes of its actions, by reasoning with the probabilities in the world model. UPTs and PCI are described in Chapter 3.

Theories are generated using a heuristic search process, guided using the agent's current sensory inputs; this search process is described in Chapter 7. The generated theories are evaluated using a Bayesian technique, described in Chapter 6, that provides a tradeoff between the accuracy and simplicity of learned theories.

Probabilistic Planning

PAGODA uses the principle of maximizing expected utility to choose its behaviors: it forward chains through the probability space of predictions, and selects the action that maximizes its expected utility. However, it does not do any metareasoning: it only plans external actions (not internal actions such as learning or searching the plan space), and always searches to a fixed search depth, determined by the designer. The planner occasionally chooses a random action instead of selecting the best apparent action, in order to ensure that exploration continues and the agent does not get stuck on a local maximum, but it does not explicitly reason about the value of taking such sub-optimal actions. The planner is described in Chapter 3.

Architecture

Figure 1.2 shows a schematic view of PAGODA. The behavior cycle of the agent is as follows:

1. PAGODA's initial learning goal is utility: that is, it will first learn theories to predict the utility of performing various actions in specified world states. The agent's utility is provided as part of its sensory input.

2. Probabilistic background knowledge[1] is used to assign a value to potential biases for each learning goal. Additionally, the current bias may be re-evaluated if the agent's best theory is not as good as it expected it to be. The bias with the highest value is sent to the hypothesis generator.

3. Sensory observations are sent to the agent by the transducer. Probabilities of old theories are updated to reflect the new evidence provided by the observations,

[1]This background knowledge is currently provided by the designer, but is potentially learnable by the agent using similar techniques to those used for learning "ordinary" knowledge.

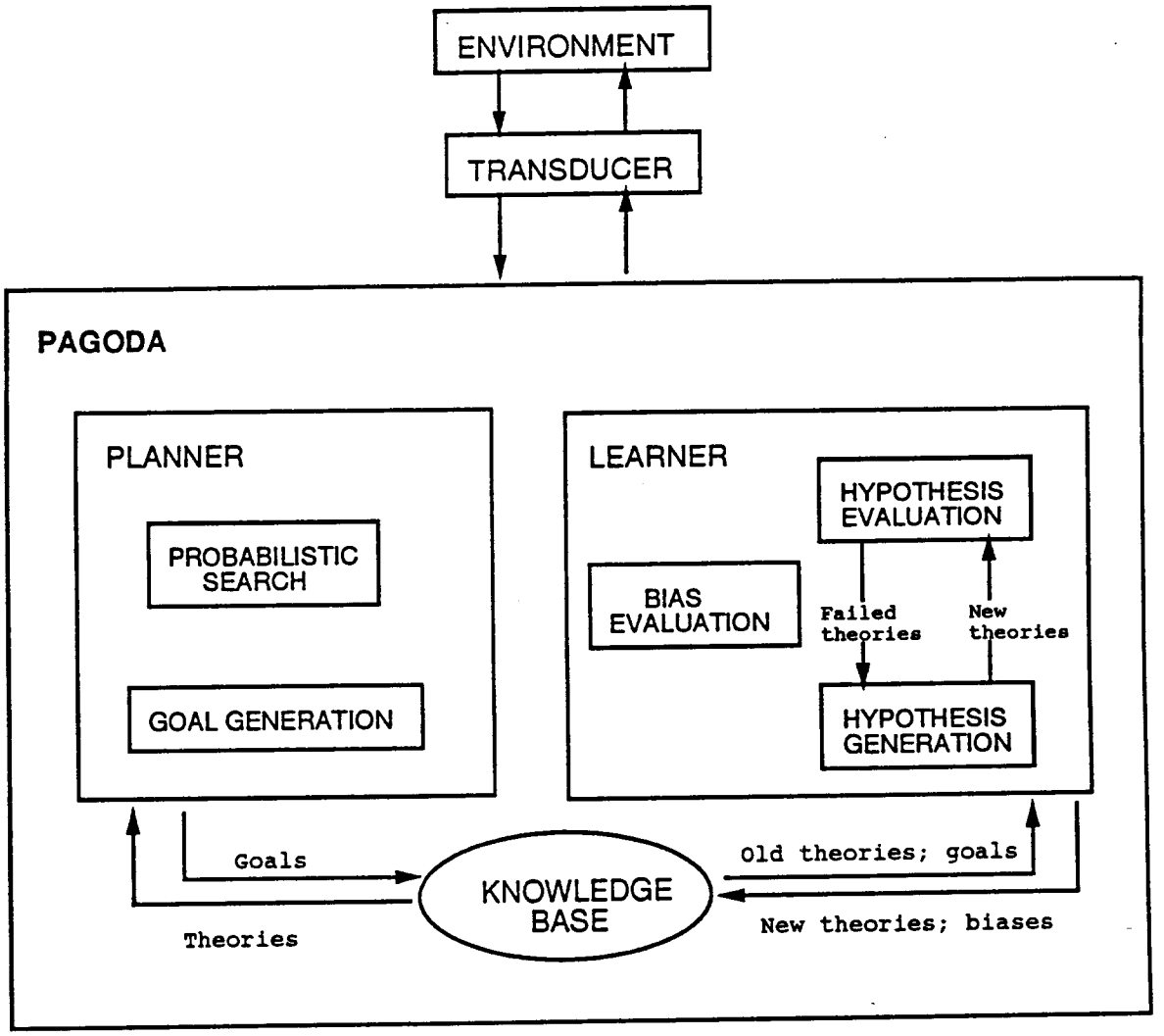

Figure 1.2: Schematic view of PAGODA

and new theories in the space defined by the bias are generated and evaluated. A set of the best theories is stored in the knowledge base.

4. The planner analyzes the preconditions of the theories to determine which features of the environment will be most useful (with respect to maximizing utility) to learn. These most useful preconditions are sent to the learner as learning goals.

5. The planner initiates a forward search through the space of possible outcomes of actions, based on the probabilistic predictions made by the current best theories. The action which maximizes expected utility is taken.

6. The action chosen by the planner is sent to the transducer, which executes the action in the real or simulated environment.

7. The sequence is repeated.

PAGODA has been implemented in the RALPH world described in Chapter 3. Empirical tests of the system are presented and analyzed in Chapter 7. The tests show that the PAGODA model works, in that the system consistently improves its predictive accuracy and average utility over time, and show the effects of the individual components on the system's performance. The tests also show that the components of PAGODA are difficult to isolate, and that they interact in nontrivial ways. We consider this to be expected and desirable in an integrated agent. Because of the tight integration, and because of the complexity of the system, however, the results are highly sensitive to initial conditions and to limitations in the design. In particular, the planning mechanism and heuristic search, which are not well developed in the current model, have an adverse effect on the system's performance.

The remainder of the thesis is organized as follows: Chapter 2 contains an overview of past work in probabilistic learning, including probability theory and maximum entropy, probabilistic logics, belief networks, and existing machine learning techniques that handle uncertainty. Chapter 3 describes PAGODA's performance element, including the representation and inference method for probabilistic theories, the RALPH world, and the planning process. Chapters 4 through 6 present

the main research results of Goal-Directed Learning, Probabilistic Bias Evaluation, and probabilistic learning. Chapter 7 describes the implementation of PAGODA in the RALPH world, and shows the results of empirical tests in various RALPH environments. Finally, Chapter 8 discusses related work, and Chapter 9 presents open research problems and conclusions.

Chapter 2

Survey of Probabilistic Learning

The problem of learning in probabilistic domains has been studied by researchers in philosophy, statistics, and artificial intelligence. This chapter provides an introductory survey of relevant research in these fields.

Section 2.1 discusses probability theory, developed by philosophers to address the problem of forming and reasoning with beliefs under uncertainty. Section 2.2 outlines two methods for representing probabilistic knowledge: a formal language combining logic and probability, and belief networks. Finally, Section 2.3 surveys methods for probabilistic learning; these include maximum entropy, Minimum Length Encoding, machine learning techniques, and planning methods for controlling the learning process.

2.1 Probability Theory

The first theories of probability were developed in the 17th century by Mill and Leibniz. Mill wanted to analyze games of chance; Leibniz was interested in analyzing legal reasoning quantitatively.[1] Despite centuries of research and study since then, there is still fundamental disagreement about what probabilities represent (or what we would like them to represent). Three different interpretations of the

[1]The interested reader is referred to [Hacking, 1975] for a detailed and fascinating history of the evolution of probability theory.

meaning of probability are commonly used:

- *Statistical* or *empirical* probability refers to the propensity of an event to occur. This propensity is presumed to be a physical property of the environment. Statistical probabilities may be directly approximated by empirical observations. For example, if we roll an unbiased die a large number of times we can estimate the statistical probability that a five will turn up as 1/6. Most standard statistical techniques deal with statistical probabilities.

- *Physical* or *logical* probability holds the relation between evidence and the probability of a hypothesis to be an objective or mathematical one. Compared to statistical probability techniques, logical probability allows for a greater range of evidence to be considered in determining the probability of a hypothesis. However, formalizing such a logic has proved to be extremely difficult. Carnap [1950] has begun to build a framework for a logic of probability, but the logic is still incomplete. Kemeny [1963] gives an example of Carnap's logic of probability applied to a simple dice-rolling problem. The probability of a specified dice roll, given a set of observations, is shown to depend on a parameter λ, representing an "index of caution" (a larger index of caution indicates that it will require more observations to change the observer's initial "unbiased" estimate of equal probability for all outcomes). Since λ must be specified by the observer, and the set of outcomes must be predefined, the resulting probabilities still depend heavily on prior beliefs as expressed in the problem statement.

- *Subjective* probability treats the relation between evidence and the probability of a hypothesis as a function of the observer. [Cox, 1946] proved that any reasoning method which represents belief as a real number, and follows certain rules of consistency, must satisfy the axioms of probability, implying a certain degree of intrinsic objectivity. However, because subjective probability is dependent on the observer, and in particular on the observer's prior probability distribution, the particular probabilities that are assigned to events will vary between individuals. Bayesian probability theory is the most familiar form of

subjective probability.

Hacking [1975] writes that

> Leibniz had learned from the law that probability is a relation between hypotheses and evidence. But he learned from the doctrine of chances that probabilities are a matter of physical propensities. Even now no philosopher has satisfactorily combined these two discoveries.

In other words, our (or at least Leibniz's) intuitive notion of probability covers both the logical and the statistical interpretations of probability, but it is not clear how they interact or whether there can be one satisfactory formal interpretation incorporating both of these views.

Non-probabilistic approaches to representing uncertainty include fuzzy logic [Zadeh, 1980], which describes uncertainty about linguistic descriptions, Dempster-Shafer theory [Shafer, 1976], in which bounds on uncertainty are maintained, and certainty factors (see, for example, [Shortliffe, 1976] and [Horvitz and Heckerman, 1986]), which are heuristic measurements of belief used by some expert systems. An introductory survey of representations for uncertainty can be found in [Wise and Henrion, 1986].

2.2 Representing Probabilistic Knowledge

In order to use probabilistic knowledge in an automated learning system, a formal system for representing and reasoning with probabilities is required. In particular, given a set of generalized conditional probabilities (i.e., a probabilistic theory) and some (possibly probabilistic) knowledge about a particular object, the system must be able to make probabilistic predictions about unobserved properties of the object.

For example, given that Chilly Willy is a penguin, that a penguin is a bird, that birds fly with probability .9, and that penguins don't fly (that is, fly with probability 0), what is the probability that Chilly Willy flies? It appears obvious that the probability is 0, but even this simple case is non-trivial to automate, and in

reality the knowledge can be much more complex. Finding an answer may involve searching through a large theory, deciding which probability or probabilities to apply, and possibly combining multiple probabilities (e.g., if Chilly Willy is a penguin who also owns a Learjet).

Kyburg [Kyburg, 1974] defined the reference class for a proposition as the features that are relevant for making probabilistic predictions about the proposition. For example, the appropriate reference class for determining whether or not Chilly Willy can fly in the previous example is the class of penguins. The reference class for a proposition will depend on what is being predicted and on what probabilities are represented in the theory or set of beliefs. Once the reference class is found, determining the probability of the proposition may require probabilistic inference from the beliefs in the theory.

Bacchus's [1990] probabilistic logic and Pearl's [1988b] belief nets provide formalisms for representing probabilistic knowledge. We discuss these two approaches in the following sections.

2.2.1 Logic and Probability

Bacchus's [1990] probabilistic logic is a formal language for representing probabilistic knowledge using first-order logic. The language provides a representation for both statistical probabilities (defined in terms of observed frequencies of events) and subjective probabilities (degrees of belief derived from the statistical probabilities). The inference mechanism provides for some manipulation of the statistical probabilities using standard axioms of probability, and for direct inference from statistical to subjective probabilities using the narrowest reference class.

The subjective probability of a proposition is given a formal interpretation as the total probability mass of all possible worlds in which the proposition is true. An example (given by Bacchus) of a subjective probability in the language is "birds fly with probability at least 0.75," written as

$$\forall x.\text{prob}(\text{bird}(x)) > 0 \rightarrow \text{prob}(\text{fly}(x)|\text{bird}(x)) > 0.75$$

The antecedent is necessary because Bacchus does not permit conditioning on a statement which is known to be false. Qualitative relationships between probabilities can also be expressed; for example, conditional independence can be explicitly written as

$$\text{prob}(A \wedge B|C) = \text{prob}(A|C)\,\text{prob}(B|C)$$

Statistical probabilities, representing frequencies of events in actual trials, have a different syntax, and require "placeholder variables" to indicate which variables are intended to vary randomly. For example, the statement "ten tosses of a coin will land heads 5 times with greater than 95% probability" is written as

$$[\text{frequency-heads}(x) = .5|\text{sequence-10-tosses}(x)]_x > 0.95 \qquad (2.1)$$

Direct inference from statistical to subjective probabilities is based on finding a statistical probability with the same reference class as the desired subjective probability. If no such probability is available, a simple type of independence is assumed non-monotonically, and the "next narrowest" reference class for which a probability is available is used. For example, if one wishes to find the probability that a particular sequence of 10 tosses of a quarter will yield five heads, and the only statistical probability available is Equation 2.1, the direct inference mechanism non-monotonically assumes independence of frequency-heads and is-quarter, given sequence-10-tosses, yielding

$$\begin{aligned}
&\text{prob}(\text{frequency-heads}(T)|\text{sequence-10-tosses}(T) \wedge \text{is-quarter}(T)) \\
&\quad = [\text{frequency-heads}(x) = .5|\text{sequence-10-tosses}(x) \wedge \text{is-quarter}(x)]_x \\
&\quad = [\text{frequency-heads}(x) = .5|\text{sequence-10-tosses}(x)]_x \\
&\quad > 0.95
\end{aligned}$$

While Bacchus's language provides a useful formalism for representing many aspects of probabilistic reasoning, including certain forms of default reasoning, it does not provide a representation for beliefs about relevance, nor does it allow default assumptions such as independence or maximum entropy to be used in the inference process.

2.2.2 Belief Networks

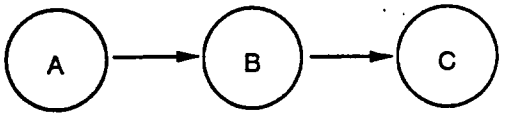

Figure 2.1: Example of a Belief Net

A belief network is a compact representation of a complete joint probability distribution on a set of propositions. Each proposition is represented as a node, and conditional probabilities (dependencies) are represented as links between nodes. Any nodes that are not directly connected are assumed to be conditionally independent, given the intervening nodes.

An example of a belief net is given in Figure 2.1. The nodes represent random variables. Arcs represent dependencies between the random variables. If there is no direct arc between two nodes, there is assumed to be no direct dependency between the random variables. For example, A and B are dependent, as are B and C. A and C, however, are conditionally independent given the intervening node B.

A probability matrix is stored at each node in the network, representing the conditional probability distribution for that node given its set of parent nodes (i.e., the nodes for which there is a direct arc from the parent to the node). For example, node B in Figure 2.1 has a table containing probabilities of the form $P(B = b_i|A = a_i)$ for each value b_i of B and a_i of A. Node A has no parent nodes, so a prior probability distribution for A is stored at the node, containing the unconditional probabilities $P(A = a_j)$.

Letting x_i stand for the event that the random variable X_i takes on value x_i, it can be shown that the joint probability distribution $P(x_1, \ldots, x_n)$ for the n nodes in a belief network depends only on the probability tables stored at each node. The joint probability is the product of the conditional probabilities of all nodes given their parents:

$$P(x_1, \ldots, x_n) = \prod_{i-1}^{n} P(x_i|\pi_i)$$

where π_i is the set of parents of node i.

One problem with belief nets as presented above is that they require a probability matrix of size $k_i \prod_{j \in \pi_i} k_j$ at every node i (where k_i is the number of values that the random variable at node i takes). Pearl [1988a] gives several models for computing this matrix from a subset of the probabilities; he refers to these models as Canonical Models of Multicausal Interaction (CMMIs). The noisy-OR model of disjunctive interaction models a set of independent causes (parents) of an event (node). Each cause has an associated "exception"— a random variable which, if true, will inhibit the effect of the cause on the event. For example, Pearl gives a situation where the event in question is a burglar alarm going off; the two causes are a burglar and an earthquake; and the two inhibitors are that the burglar is highly competent and that the earthquake has low vertical acceleration. Given an event E with Boolean causes C_i and associated exceptions with probabilities q_i, the overall probability of the event is given as:

$$P(E|c_1, \ldots, c_n) = \prod_{i:c_i is TRUE} q_i$$

This model allows the probability matrix to be computed from only n probabilities, instead of the 2^n that would be required to enumerate all of the conditional probabilities in the complete matrix.

UPTs (Uniquely Predictive Theories), the representation used by PAGODA, are a hybrid of rule-based approaches and the belief-net method of representing dependencies. UPTs consist of rules, which are easy to manipulate, perform inference with, and learn using familiar and intuitive inference rules and inductive operators. However, the rules are not modular: the semantics of a UPT does not allow the inference rules to be applied without knowing what other rules exist in the system. On the other hand, complete theories are still easy to modify incrementally because they are modular in appearance, representation, and local meaning, if not in global application.

PCI, PAGODA's inference method, provides the equivalent of a sophisticated CMMI for a node in a belief network. The probabilities stored in PAGODA's theories are used to compute the entries that would appear in the probability matrix dynamically, assuming independence where necessary.

2.3 Probabilistic Learning Techniques

In this section, we summarize some of the work that has been done in the general field of learning, or forming beliefs, in probabilistic domains.

Maximum Entropy (ME) methods, discussed in Section 2.3.1, are used by statisticians to find a "good" probability distribution, given constraints on the distribution. The ME assumption can be viewed as a bias that the generated distribution should include as little information as possible, other than the initial constraints.

Minimum Length Encoding (Section 2.3.2) derives from work by Solomonoff, Kolmogorov, and Chaitin on algorithmic complexity. The operative principle is that the length of the description of a theory, plus the length of the data encoded with respect to the theory, should be minimized.

"Traditional" machine learning approaches, including search-based learning algorithms, iterative parameter adjustment mechanisms, and genetic learning algorithms, are presented in Section 2.3.3. Finally, methods that view learning as a process of planning to collect useful data are discussed in Section 2.3.4.

2.3.1 Maximum Entropy

The Maximum Entropy (ME) principle states that given constraints on a probability distribution p, the best estimate for p will maximize the entropy function

$$H(\mathrm{p}) = \sum p_i \log p_i$$

The justification for ME is that the best distribution is the one that minimizes the amount of information, by representing only the information contained in the constraints. Entropy is an information-theoretic measure of the information in the distribution. An in-depth discussion of the justifications for ME can be found in [Hunter, 1986].

Mathematical techniques such as the Lagrange method can be used to find a distribution satisfying the constraints that maximizes entropy. (See Appendix A for an example of the Lagrange method applied to the ME constraint problem.)

The primary difficulty with ME is that the Lagrange equations can be difficult or impossible to solve directly. Approximation techniques are required if the method is to be applied automatically. This is a relatively unexplored area, but [Levine and Tribus, 1979] contains a collection of ME applications that use various domain-specific heuristics and approximation techniques.

2.3.2 Minimum Length Encoding

Solomonoff's theory of inductive inference [1964a] defines the probability of a theory, represented as a Universal Turing Machine (UTM) program, as the probability that the theory would be generated by a sequence of unbiased coin flips. If T is the length of a theory, then the probability of the theory is

$$P(T) = 2^{-|T|}$$

The most probable theory to explain a set of data is the shortest theory that generates that theory as output. The probability of the next observation taking on a given value is the sum of the probabilities of the theories that predict that value.

Chaitin [1977] extended Solomonoff's theory, developing a more formalized concept of algorithmic complexity. Rissanen's stochastic complexity [1987] also includes structure-dependent terms (for example, the number of parameters of a theory), allowing the model class to be selected automatically. Essentially, the model class is included in the description length, so that simpler (shorter) classes are automatically preferred.

MLE applications generally require a hand-tailored encoding scheme for the particular domain. In other words, the designer must decide what the description language will be, and therefore what theories can be represented, and how many bits are required to describe any given theory.

Pednault [1989] applies the MLE principle to surface reconstruction. The system works quite well, but the data-encoding method is hand-tailored for the application. A methodology for applying MLE is outlined: (1) determine structures to be detected, (2) develop language, (3) develop algorithms, (4) run tests, (5) fix errors

and iterate. This methodology assumes a significant amount of human intervention: only the tests run in step (4) are automated.

Babcock [1990] describes an application of MLE to the analysis of DNA sequences. The goal is to segment DNA into functional regions, which can then be matched to corresponding RNA features, and finally to functional protein features. The method involves constructing an encoding, using a statistical Markov model, for the DNA sequence. Again, the encoding is domain-specific, but the application demonstrates the utility of the MLE model for extracting regularities in any type of data.

A great deal of domain knowledge is embodied in the encoding procedure for these applications; if a more general method for expressing the domain knowledge could be found, and used to derive an encoding automatically, MLE could be applied with much less human effort.

2.3.3 Probabilistic Machine Learning

Quinlan was one of the earliest machine learning researchers to consider seriously the effect of noise on concept learning [Quinlan, 1986]. He extended ID3 (a decision-tree learning algorithm [Quinlan, 1983]) to learn decision trees under noisy conditions by adding a chi-square test for independence: if the distribution of positive and negative instances with varying attribute values is (approximately) equal to that expected from randomly assigned classifications (i.e., the attribute values are statistically independent of the classification value), then the decision tree should not be split at that point.

Quinlan's approach suffers from some limitations: first, it learns decision trees, which are difficult to impose certain syntactic learning biases on (such as a limited number of disjunctions, or relational descriptions). Second, incremental versions are not adequate for complex domains: Utgoff's ID5 [1988] is fairly expensive to use, and it is not clear how good the trees it generates will be on average.

Schlimmer's STAGGER [1987a] represents concepts as prototypes. Individual features are maintained with associated sufficiency and necessity statistics; these

statistics are combined to make predictions about new instances. New features are formed as conjunctions and disjunctions of existing features using an *ad hoc* heuristic process. The primary problems with the approach are that the representation can be difficult for a human user to interpret and that features are assumed to combine independently.

Goodman's ITRULE [1989] learns sets of probabilistic rules. The rules generated are the K "most informative" rules, where K is a parameter provided by the designer, and "informativeness" is based on an information-theoretic measure of the rule's content. It is not clear how useful it is in practice to learn a fixed number of independently informative rules.

Rendell's PLS [1986], a system that builds hyperrectangles in the instance description space to describe learned concepts, is basically a simple statistical technique for finding "good" hyperrectangles. A variety of heuristic techniques are used to generate and evaluate hyperrectangles. These heuristics are not applied in a coherent framework, though. Additionally, the use of hyperrectangles as the representation limits the expressivity of the learned theories.

CONSTRUCTOR [Fung and Crawford, 1990] is a technique for building Markov networks (essentially belief nets with undirected arcs) from data. CONSTRUCTOR attempts to find the best structure to represent the dependencies in the data. A chi-square test for independence is used to find a set of neighbor nodes for each node in the net; these neighbor nodes "shield" the node from the influence of other nodes in the net (in other words, the node is conditionally independent of the remaining nodes in the net, given its neighbor nodes).

Cooper and Herskovitz [1991] describe a method for building belief nets that is based on a Bayesian evaluation technique. Assuming a uniform prior distribution on belief net structures and an ordering on the variables in the belief net yields a complex formula for the probability of a structure. A good structures is found using a greedy method: the best parent of a node that increases the overall probability of the structure is added at each step, until no such parents remain.

Holland's genetic learning algorithms [1986] are used to form many independent rules which compete in a fashion inspired by Darwinian evolutionary theory.

Rules reproduce, mutate, and are combined to generate new rules. As in PLS, the rule-generation operators have an *ad hoc* flavor to them. Additionally, the learned theories are potentially very difficult for an observer to interpret, since there are many independent rules with no central control or inference method.

Buntine's work on Bayesian learning [1990] analyzes the problem of learning probabilistic classification rules as a search problem, and gives guidelines for formally analyzing such search algorithms. Buntine also describes a Bayesian method for learning class probability trees, based on previous work on learning decision trees (e.g., [Quinlan, 1986]) but using Bayesian, rather than information-theoretic, techniques for splitting trees and averaging predictions over multiple trees.

2.3.4 Planning to Learn

Doyle's [1990] definition of learning is "interpreting experience by making rational changes of mental state or expectation." Being rational means deciding whether and what to learn based on the expected utility gain of doing so (due to the increased accuracy of predictions) and the associated cost of learning, storing, and applying the learned theories to maximize utility. Choices that must be made include: which concepts to learn, what relevance criteria to use, which apparent distinctions are significant, what experiments to run, how much evidence to collect, and which conclusions or assumptions should be preferred. Determining the gains and costs of learning in order to make these choices is a difficult problem that has not been addressed extensively in the literature.

Subramanian [1986] gives a method for generating discrimination experiments for a version-space learning algorithm. The version space is factored into independent relations; this allows the version space to be expressed as several smaller, independent version spaces. The discrimination experiments are instances which divide the remaining version spaces as nearly in half as possible. This allows the learner to reduce the number of potential hypotheses in half after each instance.

The Operator Refinement Method is used in [Gil, 1991] to identify experiments which allow the system to refine an incorrect theory. When the theory fails

(makes an incorrect prediction), the system generates a set of preconditions that might account for the failure, and generates experiments to identify the correct precondition. The approach, which has been implemented in PRODIGY, assumes a completely deterministic world.

The Map-Learning Critter [Kuipers, 1985] and Rivest and Schapire's [1987] method for learning deterministic finite-state automata both perform deliberate exploration of their environments to learn a world model, by generating sequences of experiments to refine the existing model. These systems are discussed in Section 8.3.

Rivest and Sloan [1988] model the process of inductive inference as a tradeoff between "thinking" and "doing." The costs of making predictions and of doing experiments are assumed to be constant. Bayesian updating is used to assign probabilities to theories, given some prior distribution and a sequence of observations (results of experiments). Given current probabilistic beliefs in a set of possible theories, the method determines a sequence of actions (chosen from a finite set of choices) that maximizes the rate of progress with respect to one of five optimization criteria. An example of an optimization criterion is to maximize the expected total probability mass associated with theories which will be refuted by an action sequence. The primary limitation of the model is that the theories are simple deterministic predictive functions, and it is assumed that a correct theory exists.

The n-armed bandit problem addresses the problem of experiment generation in nondeterministic environments. The problem is formalized as follows: given a slot machine with n arms, and some state of knowledge about the probability of success associated with each arm, what sequence of actions (arm pulls) maximizes the expected rate of success? Given perfect knowledge, a rational agent should always pull the arm with the highest expected rate of success. However, given only partial knowledge (i.e., a set of observations providing some current estimate of the rates of success), the problem becomes more difficult. If the agent uses the policy of always pulling the arm with highest estimated probability of success, it can easily be misled by an incorrect initial estimate into preferring an arm with relatively low actual probability of success. An optimal policy should gather enough information to converge on the best arm in the long run, while maximizing expected success dur-

ing the information-gathering stage. Berry [1985] surveys solutions to the n-armed bandit problem for a variety of initial conditions and independence assumptions.

Chapter 3

Performance Element

This chapter describes the components of PAGODA's performance element. These components include the representation and inference method for probabilistic theories, and the probabilistic planning technique. A shorthand notation for theories is also described.

The next section describes the RALPH testbed; we will use examples from the RALPH world throughout the thesis the illustrate the components of PAGODA.

3.1 RALPH: An Autonomous Agent Testbed

RALPH is a system developed at UC Berkeley as a testbed for designing intelligent autonomous agents. RALPH, which runs on TI Explorers in ZetaLisp and on DECstations in Allegro Common Lisp, is an object-oriented system with scheduling software and a graphic display, and is designed to be easily extensible. The system provides the infrastructure for designing, running, and testing new worlds and agents.

This section provides an overview of the capabilities and use of the RALPH system. For further details, see [Parr *et al.*, 1992]; this document and the RALPH software can be obtained by sending electronic mail to `ralph@guard.berkeley.edu`.

RALPH's time-slicing mechanism simulates an asynchronous world by running the agents in pseudoparallel: it allocates a fixed amount of time (called a "time slice" or "tick") sequentially to each agent (note that this allows the RALPH world

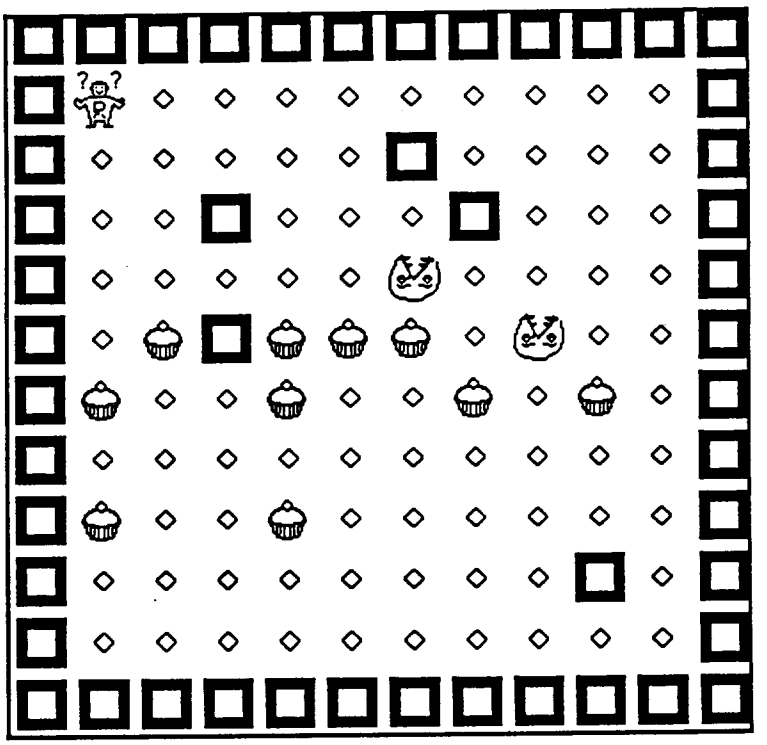

Figure 3.1: Typical RALPH world

to be used for testing multiple-agent interactions) and to the world. Because of this, the agent's activity and the world's behavior are interleaved, creating a realistic simulation of an embedded agent. However, time is necessarily discretized, making continuous processes difficult to represent.

RALPH's versatility as a testbed comes from the object-oriented design of the world model and the agent model. There are currently several worlds implemented (including the nasty world described below, a traffic-crossing world, and a space-invaders world), but it is simple to add new objects or processes or to define a completely new world by defining new flavors and methods. New ralphs[1] can be designed by defining methods on top of existing default agents. The agent (including the sensory inputs) and nasty world we describe below can be replaced with other agents and worlds.

A typical RALPH world is shown in Figure 3.1. The objects in this world are ralph (the protagonist, with an "R" on his chest and question marks over his head), nasties (the antagonists: evil-looking creatures with large, sharp teeth), food (cupcakes) and walls. Empty locations are represented by diamonds. In the PAGODA implementation, there is only one ralph (PAGODA-RALPH, hereinafter referred to as PR).

The object hierarchy in RALPH includes agents and other world objects. Some world objects, such as walls, food, and nodes, are basically static; their properties include a location (`x` and `y` coordinates), appearance, and size. Objects also have various manipulation methods (e.g., food has a `:get-eaten-by` method). Agents (ralphs and nasties) are dynamic: their `:perform` methods are run at each time slice to update their state. Additionally, they have properties that are maintained by the world (such as location and appearance), sensory inputs, and actions that they can perform. World objects generally change only when agents manipulate them, but worlds can be designed in which objects appear, disappear, or move randomly.

Walls are immovable, and ralphs cannot move into a space containing a wall. Food (which is represented by a cupcake icon) is scattered about the world; when a

[1] "RALPH" refers to the system; "ralph" or "a ralph" refers to an agent in the RALPH world.

ralph consumes food, its utility goes up. Nodes are special world objects, representing the discrete points in the world (i.e., grid locations). Nodes can contain things and have `:move-into` methods.

The world is updated at each time click as follows: the `:start-slice` method of each agent in the world is run; the `:perform` methods are run for a fixed time interval;[2] the `:end-slices` are run to perform any necessary clean-up actions; and finally each agent's `:choose-action` method is run. The world is updated by applying the selected actions of each agent. Currently, all of the agents' actions have deterministic outcomes given the complete state of the world, but actions with non-deterministic effects (e.g., movement actions with a probability of error) could be written.

Nasties exhibit fairly simple behavior: if they can't see a ralph, they turn or move randomly; if they can see a ralph but aren't adjacent to it, they move towards it; if they can see a ralph adjacent to them, they bite it.

PR is implemented as a ralph that calls a learning routine after every time slice (in the `:end-slice` method), and uses a probabilistic planning mechanism in the `:choose-action` method to select actions based on the predictions of its learned world model. The actions available to PR are `:move-forward`, `:turn-left`, `:turn-right`, `:munch`, and `:zap`. The effects of these actions are as follows:

`:move-forward`: If the space in front of PR contains only food, or is empty, it moves forward and loses 10 utility points. If the space contains a wall or a nasty, PR bumps into it and loses 11 utility points.

`:turn-left`: PR turns 90 degrees left and loses 10 utility points.

`:turn-right`: PR turns 90 degrees right and loses 10 utility points.

`:munch`: If there is food in the space containing PR, it eats part of the food and gets 90 points (100 points for eating minus 10 points for the energy spent eating). Otherwise, it loses 10 points. Each cupcake takes three munches to finish.

[2] PAGODA's `:perform` method is not actually time-sliced: it is allowed to run to completion.

`:zap`: If there is a nasty in the space in front of PR, the nasty disappears and PR loses 160 points, else it loses 10 points.

If a nasty is next to and facing PR, it will bite the agent, causing it to lose 50 utility points (in addition to whatever effect PR's actions have on its utility).

PR has four sensory inputs. `nasty-smell` and `food-smell` are sums over all of the objects in the world of their smell intensity, which is inversely proportional to their distance from PR. `vision` has two arguments, the type of the nearest object directly in front of PR and its distance. This object may be a `wall`, `nasty`, or `food`; PR also knows that walls and food are `inanimate-objects`, and can use this term in building theories. Δu is the change in the agent's utility function at each click. Other ralphs with more sophisticated sensory inputs have been built: for example, the `vision` input in some ralphs consists of a set of objects and their apparent sizes and angles.

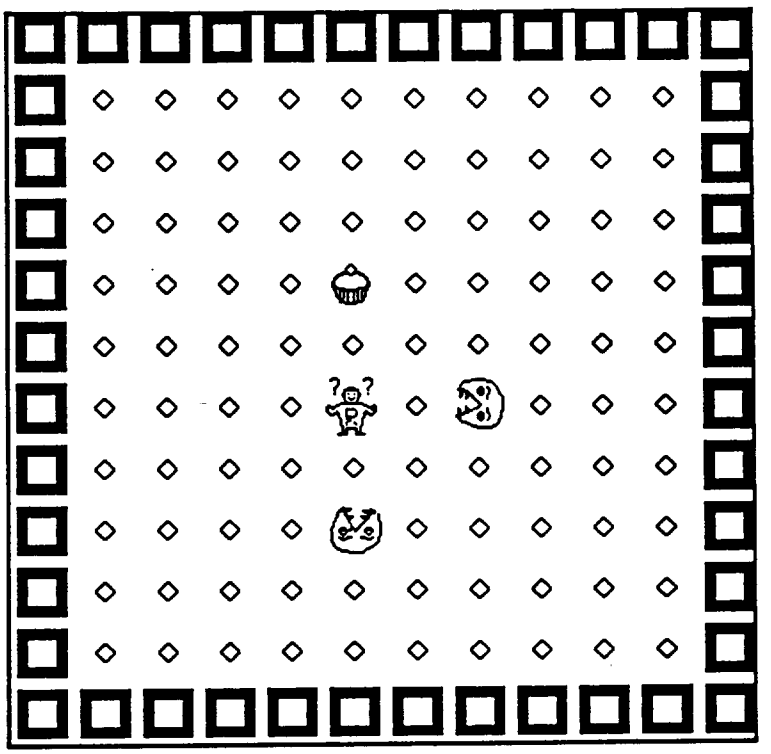

Figure 3.2: RALPH's "nasty world"

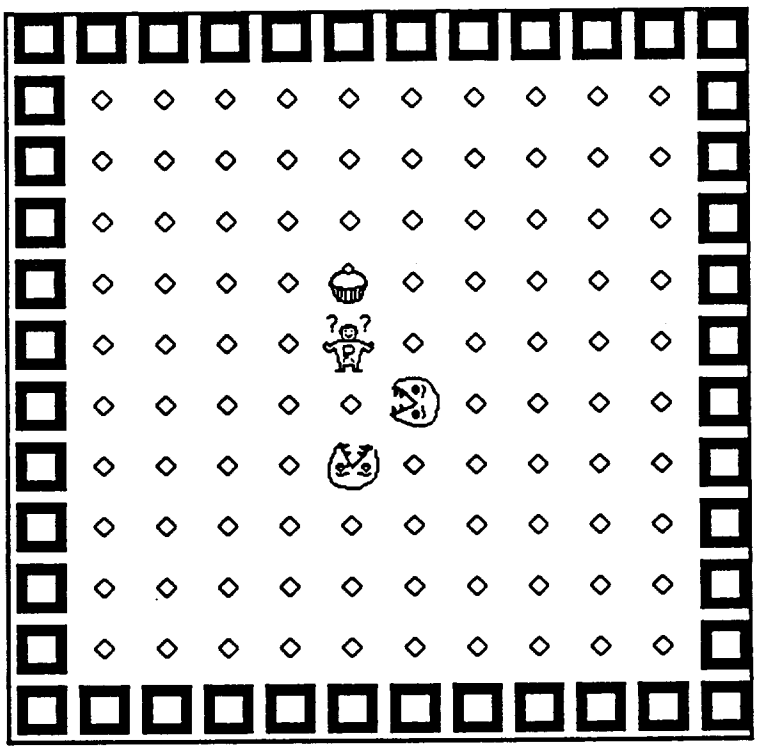

Figure 3.3: Nasty world at the next tick

In Figure 3.2, PR's current sensory input consists of the sentence

$$\texttt{vision}(10, \texttt{food}, 2) \wedge \texttt{food-smell}(10, 5) \wedge \texttt{nasty-smell}(10, 10) \wedge \Delta u(10, -10)$$

The first argument of each predicate represents the current time. PR's best bet would be to move forward (toward the food), yielding at the next tick

$$\texttt{vision}(11, \texttt{food}, 1) \wedge \texttt{food-smell}(11, 10) \wedge \texttt{nasty-smell}(11, 10) \wedge \Delta u(11, -10)$$

The new situation is represented in Figure 3.3. Notice that the time is now 11, the food smell is stronger, and the nasties have moved towards PR's former position.

3.2 Representing Probabilistic Knowledge

PAGODA's knowledge is represented as probabilistic theories about features of the world (i.e., learning goals). Each theory consists of a set of conditional probability distributions; each of these specifies the observed distribution of values of the goal, given the conditioning context. Conditioning contexts consist of a perceived world and possibly an action taken by the agent. A probabilistic inference mechanism is used to make predictions about the effect of the agent's action on its learning goals, given the agent's perceptions (which may be the current perceived world, or a hypothetical perceived world generated by the planner). This mechanism requires determining which conditional distributions within a theory are relevant, and combining them if necessary (using minimal independence assumptions) to get a single predicted distribution.

The theories are called uniquely predictive theories (UPTs) because PAGODA imposes a restriction on the structure of the theories that allows the inference mechanism to find a unique predicted distribution for any perceived world. The building blocks of UPTs are presented in the next three sections: Section 3.2.1 introduces conditional probability, conditional distributions are defined in Section 3.2.2, and

Section 3.2.3 defines predictive theories. UPTs are a subset of predictive theories; they are described in Section 3.2.4.

A shorthand notation for theories is given in Section 3.3; this notation is used throughout the thesis. The inference mechanism, called PCI (Probability Combination using Independence), is described in Section 3.4; it is used to compute the likelihood of a theory (Section 6.3) and to make predictions for planning (Section 3.5).

3.2.1 Conditional Probability

Definition: The **conditional probability** or **CP** of X given Y is

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)}$$

X, the **target**, and Y, the **conditioning context** or **CC**, are first-order schemata. These schemata are required to be conjunctions of feature specifications, where each feature specification may contain internal value disjunctions, representing internal nodes in a feature value hierarchy (for example, $\texttt{vision}(t, \texttt{wall} \vee \texttt{food}, [1,3])$ means that at time t, the agent sees a wall or food between 1 and 3 nodes away). The schema corresponds to a set of perceived worlds. For example, the following is a valid schema in the RALPH world:

$$\texttt{vision}(t, \texttt{any-object}, 1) \wedge \texttt{nasty-smell}(t, [10, \infty]) \wedge \Delta\texttt{u}(t, -100)$$

Cross-feature disjunctions, such as $\texttt{vision}(t, \texttt{food}, 2) \vee \texttt{food-smell}(t, 20)$, are not allowed. Negations of features are allowed, since they may be rewritten as disjunctions.

An example of a conditional probability is

$$P(\Delta\texttt{u}(t+1, -10) | \texttt{action}(t, \texttt{:move-forward})) = .75 \qquad (3.1)$$

The variable t stands for any time at which this conditional probability is the most specific in the theory; that is, the knowledge we have about the situation at time t implies this CC and does not imply any other more specific CC. Variables are *not* universally quantified, since they cannot be instantiated without examining the rest of the theory. The semantics of any individual probability within a theory will therefore

depend on the content of the rest of the theory, as well as on the inference mechanism used to instantiate the variables and make predictions.

Intuitively, the meaning of Equation 3.1 is: given that an agent executes the action `:move-forward` at time t—and that is all the relevant information the agent has—the probability that the agent's change in utility at time $t+1$ will be -10 is .75. The information in the conditioning context is assumed to be the only relevant information if the agent have no other CP with a more specific conditioning context. For example, if the agent also knows that $\texttt{vision}(t, \texttt{wall}, 1)$ holds, and the theory contains the CP

$$P(\Delta \mathrm{u}(t+1, -10) | \texttt{action}(t, \texttt{:move-forward}) \wedge \texttt{vision}(t, \texttt{wall}, 1)) = 0$$

this more relevant conditional probability will be used (and Equation 3.1 has no bearing on the prediction the agent makes). On the other hand, the agent may have more knowledge about t—such as the fact that `nasty-smell` was 0—that is not mentioned in any CP in the theory; this information is considered to be irrelevant in the context of the current theory.

Using conditioning, relevance, and specificity in this way yields a quasi-non-monotonic representation: adding new knowledge (i.e., new conditional probabilities) to a theory doesn't change the *truth* of the rest of the CPs in the theory, but it may change their range of applicability, and therefore change their semantics.

3.2.2 Conditional Distributions

> **Definition: A conditional distribution or CD,** which we will usually refer to as a **rule,** is a set of n conditional probabilities on a target schema G (a learning goal), with mutually exclusive partial variable substitutions $\theta_1 \ldots \theta_n$ and common conditioning context C, such that
>
> $$\sum_{i=1}^{n} P(G\theta_i | C) = 1$$

A CD specifies all of the possible instantiations for a target given a particular context, and their probabilities. If C contains all of the relevant information, this distribution is used to predict the probability of each value of G.

For example, PR's utility goes up when it does a :munch action, but only if there is food in the same node. If this has been true on half of the occasions it's tried a :munch action, it may have a CD on $\Delta\mathrm{u}(t1, du)$ containing the probabilities

$$P(\Delta\mathrm{u}(t+1, 90)|\mathtt{action}(t, \mathtt{:munch})) = .5$$
$$P(\Delta\mathrm{u}(t+1, -10)|\mathtt{action}(t, \mathtt{:munch})) = .5 \quad (3.2)$$

This is a rule on goal $\Delta\mathrm{u}(t1, du)$ with substitutions $\theta_1 = \{t1/t+1, du/90\}$ and $\theta_2 = \{t1/t+1, du/-10\}$.

A CD with an empty conditioning context is referred to as a prior distribution on G, or a **default rule** for G. A **prediction** on G is the set of probabilistic outcomes specified by a conditional distribution. The rule in Equation 3.2 makes the prediction

$$\{(\Delta\mathrm{u}(t+1, 90), .5), (\Delta\mathrm{u}(t+1, -10), .5)\}$$

3.2.3 Predictive Theories

> **Definition:** A **predictive theory** or **PT** on goal schema G is a set of m conditional distributions, or rules, on G, with conditioning contexts $C_1 \ldots C_m$ (which must be distinct but not necessarily disjoint), such that any *situation* (consisting of a perceived world and possibly an action) implies at least one of the conditioning contexts.

As long as a set of distinct rules on a goal schema includes a default rule it is guaranteed to be a predictive theory.

A predictive theory stores all of the beliefs the agent has about the goal G. The rules in a theory are indexed by their conditioning contexts (i.e., the situations in which they apply). Using a specificity relation between CCs, the rules can be organized into a DAG in which a child is always more specific than its parents. (A rule in the theory may have multiple parents, but no rule may be an ancestor of itself.)

Figure 3.4 shows an example of a predictive theory on a goal G, drawn as a DAG. Only the conditioning contexts are shown, indicating the structure of the theory. A conditional distribution is actually stored at each node. For example,

the bottom node represents a rule containing conditional probabilities of the form $P(G\theta_i|A(x) \wedge B(x)) = p_i$.

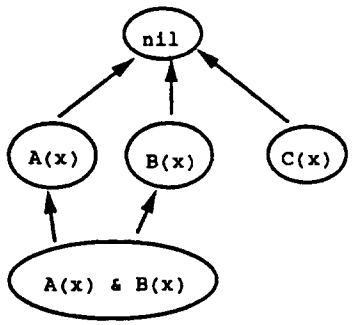

Figure 3.4: Example of a predictive theory

> **Definition:** The most specific rules for a perceived world S are the rules in the theory whose conditioning contexts are more general than S, such that no more specific rule's conditioning context is also more general than S.

A rule is an MSR for a situation S if its CC is more general than S and it has no children whose CCs are also more general than S. The MSRs for S are the conditional distributions that will be used to make predictions about the outcome of the specified action in the world state. If there is only one MSR, the outcome predicted by that rule is made. However, if there are multiple MSRs, their predictions must be combined. Uniquely Predictive Theories, described in the next section, are a restricted form of predictive theories that allow MSRs to be combined using simple independence assumptions; the inference mechanism which does this is described in Section 3.4.

3.2.4 Uniquely Predictive Theories

A predictive theory may correspond to many different complete probability distributions. In principle, probabilities that are not specified by the theory may take on any value that is consistent with the probabilities in the theory. For example, given only the probabilities in Figure 3.4, $P(G|D(x))$ may take on any value. In order to make predictions about perceived worlds that are not explicitly mentioned

as CCs in a rule in the theory, a single distribution must be found that specifies the remaining probabilities.

The Maximum Entropy (ME) principle, discussed in Chapter 2, provides one method for finding a "best" distribution, using the rules in a theory as constraints on the distribution. The distribution chosen using this method will add the least information possible to the existing theory. However, in the general case (i.e., for arbitrary constraints), ME is intractable.

A less expensive approach is to identify valid independence assumptions and use them to find the joint distribution. We restrict the set of allowed theories so that a unique distribution can be found using only simple independence assumptions that are consistent with the theory. If the induction mechanism finds a theory that contains all dependencies that actually exist and no others, it is safe in the limit (by definition) to assume that any dependence not represented in the agent's theory does not exist. PAGODA's Bayesian evaluation technique will discard any theory that contains additional dependencies (irrelevant rules) in favor of a simpler theory without the irrelevant rules; similarly, any theory that is missing dependencies that actually exist (i.e., statistically significant correlations in the data) will be discarded for one that the dependencies.

PAGODA's inference mechanism is based on the independence-assumption approach. The technique involves finding shared features in the conditioning contexts of rules to be combined (MSRs), and assuming that the remaining features are independent, given the shared features.

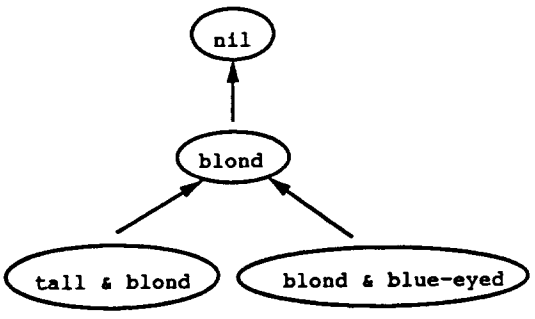

Figure 3.5: Sample UPT

Suppose our theory (shown in Figure 3.5) specifies the conditional distributions corresponding to

$$P(\texttt{Swedish}(x)|\texttt{tall}(x) \wedge \texttt{blond}(x))$$

$$P(\texttt{Swedish}(x)|\texttt{blond}(x) \wedge \texttt{blue-eyed}(x))$$

Now we observe someone who is tall, blond, and blue-eyed. Given our theory, we wish to find the probability that they are Swedish, i.e.,

$$P(\texttt{Swedish}(x)|\texttt{tall}(x) \wedge \texttt{blond}(x) \wedge \texttt{blue-eyed}(x))$$

If we assume that `blue-eyed` and `tall` are independent, and that they are conditionally independent given `blond`, this can be rewritten as

$$\frac{P(\texttt{Swedish}(x)|\texttt{tall}(x) \wedge \texttt{blond}(x))P(\texttt{Swedish}(x)|\texttt{blond}(x) \wedge \texttt{blue-eyed}(x))}{P(\texttt{Swedish}(x)|\texttt{blond}(x))}$$

`blond` is the shared feature of the two CCs, which is used to separate their effects. The denominator represents the combined effects of the two rules; the numerator represents the overlap (essentially the shared part of the world state that was included twice). If $P(\texttt{Swedish}(x)|\texttt{blond}(x))$ were removed from the theory, we would assume that it was equal to the prior $P(\texttt{Swedish}(x))$ (i.e., the MSR for $\texttt{blond}(x)$).

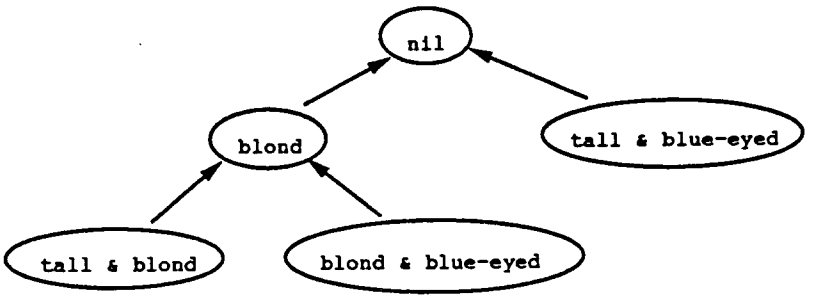

Figure 3.6: Unacceptable UPT

However, if we add the distribution specifying

$$P(\texttt{Swedish}(x)|\texttt{tall}(x) \wedge \texttt{blue-eyed}(x))$$

yielding the theory shown in Figure 3.6, we would need to assume that `blond`, `blue-eyed`, and `tall` were all independent. But if they were, this wouldn't be the

simplest theory: a perfect induction mechanism would have preferred the theory shown in Figure 3.7.

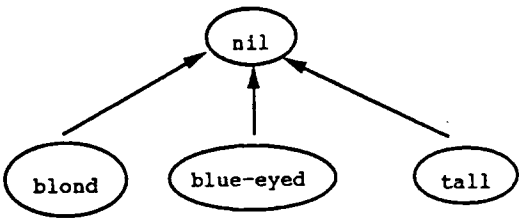

Figure 3.7: Preferred UPT

The inference mechanism does not work on theories such as the one in Figure 3.6, which have interlinked dependencies such that independent features cannot be pulled out individually. This intuition is formalized below.

> **Definition:** A set of rules in a theory is a **valid set of MSRs** if it corresponds to some situation; i.e., there must be situation (perceived world plus an action) that would have the set of rules as its MSRs.

In Figure 3.4, invalid sets of MSRs include $\{\texttt{nil}, \texttt{C}(x)\}$ ($\texttt{C}(x)$ should be the only MSR) and $\{\texttt{A}(x), \texttt{B}(x)\}$ (since $\texttt{A}(x) \wedge \texttt{B}(x)$ would be a valid MSR for the situation).

> **Definition:** The **shared features** of a set of rules are the features that appear in all of the conditioning contexts and have some value in common.

"Shared features" may also refer to this shared set of values for the features, in which case they may be thought of as the minimum specializations of the common features. For example, the shared feature of $\texttt{blond}(x) \wedge \texttt{blue-eyed}(x)$ and $\texttt{tall}(x) \wedge \texttt{blue-eyed}(x)$ is $\texttt{blue-eyed}(x)$. The shared features of

$$\texttt{vision}(10, \texttt{food}, 2) \wedge \texttt{food-smell}(10, 5) \wedge \texttt{nasty-smell}(10, 10) \wedge \Delta\texttt{u}(10, -10)$$

and

$$\texttt{vision}(t, \texttt{food}, [1, 3]) \wedge \texttt{action}(t, \texttt{:move-forward}) \wedge \texttt{food-smell}(t, 20)$$

are

$$\texttt{vision}(t, \texttt{food}, 1) \wedge \texttt{action}(t, \texttt{:move-forward})$$

Definition A set of rules is **separable** if there is some rule in the set (which is also referred to as separable with respect to the rest of the set) whose conditioning context can be split into two parts: one group of features that is shared with a single other rule in the set, and one group of features that is shared with no other rule in the set. Either group of features may be empty.

The restriction on UPTs is simply that every valid set of MSRs must be separable. Figure 3.6 violates this restriction because the valid set of MSRs

$$\{\texttt{tall}(x) \wedge \texttt{blond}(x), \texttt{blond}(x) \wedge \texttt{blue-eyed}(x), \texttt{blue-eyed}(x) \wedge \texttt{tall}(x)\}$$

is not separable: all of the rules in the set share features with both of the other rules.

3.3 Theory Notation

A shorthand notation for theories is used throughout the thesis. Rules are represented as implications with attached probabilities. They should *not* be interpreted as logical implications, but as conditional probabilities. For example,

$$\texttt{action}(t, \texttt{:munch}) \rightarrow_{.6} \Delta\text{u}(t+1, 90)$$

represents the conditional probability

$$P(\Delta\text{u}(t+1, 90) | \texttt{action}(t, \texttt{:munch})) = .6.$$

The $\Box$ symbol is used to indicate an empty conditioning context. Repeated conditioning contexts in a single rule are left out for readability. An example of a UPT in this notation is:

$$\begin{array}{ll} \Box & \rightarrow_{.7} \Delta\text{u}(t+1, -10) \\ & \rightarrow_{.3} \Delta\text{u}(t+1, -60) \\ \texttt{action}(t, \texttt{:munch}) & \rightarrow_{.6} \Delta\text{u}(t+1, 90) \\ & \rightarrow_{.3} \Delta\text{u}(t+1, -10) \\ & \rightarrow_{.1} \Delta\text{u}(t+1, -60) \end{array}$$

3.4 Probabilistic Inference

This section describes Probability Combination using Independence (PCI), the inference method that is applied to a UPT T to compute the distribution of T's goal G, given a situation S. Given a set of MSRs, PCI iteratively finds a separable rule in the set, computes its contribution to the overall probability using independence assumptions, and recurses using the remaining rules as the new set of MSRs to explain the remaining features. The algorithm operates as follows:

1. Let R be the set of the n most specific rules (MSRs) in T that apply to S. This set consists of all rules, r_i, whose conditioning context C_i is satisfied by the situation, where no strictly more specific rule also satisfies the situation:

$$R = \{r_i : [S \rightarrow C_i] \wedge \neg\exists r_k, k \neq i : [(S \rightarrow C_k) \wedge (C_k \rightarrow C_i)]\}$$

2. The rules are ordered so that each rule r_i is separable given the set of rules $r_{i+1}, \ldots, r_n$. Recall that r_i is separable with respect to a set of rules if its conditioning context can be split into two parts: f_i^s, a group of features (possibly empty) that is shared with some rule in the set, and f_i^u, the remaining features, which are shared with no other rule in the set (i.e., are unique to r in this set of rules). This is guaranteed to be possible if T is a UPT, since each set of rules $r_i \ldots r_n$ is a valid set of MSRs.

3. The probability of $G\theta$ is computed, for each θ common to all rules in the set of MSRs (i.e., for values of G that are assigned non-zero probability by every rule in the set). If we assume that f_i^u is independent of the features only found in the rest of the rules (i.e., of $\bigcup_{k>i} C_k - f_i^s$), and also conditionally independent of those features given G and f_i^s (yielding a total of $2(n-1)$ independence assumptions, all consistent with the dependencies explicitly expressed in the theory), this probability is equal to

$$P(G\theta|S) = \frac{\prod_{i=1}^{n} P(G\theta|C_i)}{\prod_{j=1}^{n-1} P(G\theta|f_j^s)} \tag{3.3}$$

(The derivation of this equation is given below.) If n is 1, the product in the denominator is defined to be 1, so the predicted distribution on a goal when only one rule applies is simply the distribution given by that rule.

4. The probabilities in the denominator of Equation 3.3 are computed by applying PCI recursively.

The resulting probabilities are derived probabilities, which may be used to make further inferences in the planning process, but otherwise are not reused. Specifically, they are not stored in the theory. This keeps the empirical probabilities represented in the theory distinct from the inferred, subjective probabilities (they are subjective because the independence assumptions have not been directly validated against the data).

The formula given in Equation 3.3 is derived as follows. Consider the effects of pulling out the first MSR, r_1, and assuming that its unique features f_1^u are independent of the remaining features ($\bigcup_{j>1} C_j - f_1^s$), and independent of these features given G and f_1^s. In order to simplify the derivation somewhat, we assume that r_2 is the rule that shares the feature f_1^s. This is not necessarily the case: in fact, r_2 is simply the next separable rule. However, making this assumption does not affect the validity of the derivation. We will refer to the features in r_2 that are not shared with r_1 as f_2^r. Then using only Bayes' rule[3]

$$
\begin{aligned}
P(G|S) &= P(G|f_1^u \wedge f_1^s \wedge f_2^r \wedge C_3 \ldots C_n) \\
&= \frac{P(G|f_1^s)\, P(f_1^u \wedge f_2^r \wedge C_3 \ldots C_n | G \wedge f_1^s)}{P(f_1^u \wedge f_2^r \wedge C_3 \ldots C_n | f_1^s)} \\
&= \frac{P(G|f_1^s)\, P(f_1^u|G \wedge f_1^s)\, P(f_2^r \wedge C_3 \ldots C_n | G \wedge f_1^s)}{P(f_1^u|f_1^s)\, P(f_2^r \wedge C_3 \ldots C_n | f_1^s)} \\
&= \frac{P(G|f_1^s) \frac{P(f_1^u|f_1^s)\, P(G|f_1^u \wedge f_1^s)}{P(G|f_1^s)} \frac{P(f_2^r \wedge C_3 \ldots C_n|f_1^s)\, P(G|f_2^r \wedge C_3 \ldots C_n \wedge f_1^s)}{P(G|f_1^s)}}{P(f_1^u|f_1^s)\, P(f2 \wedge C_3 \ldots C_n | f_1^s)}
\end{aligned}
$$

[3]This is a slightly non-standard version of Bayes' rule. The general form of the rule we use here is:

$$P(X|Y \wedge K) = \frac{P(X|K)\, P(Y|X \wedge K)}{P(Y|K)}$$

$$= \frac{P(G|f_1^u \wedge f_1^s)\, P(G|f_1^s \wedge f_2^r \wedge C_3 ... C_n)}{P(G|f_1^s)}$$
$$= \frac{P(G|C_1)\, P(G|C_2 ... C_n)}{P(G|f_1^s)}$$

Iterating on the last term in the numerator yields Equation 3.3.

If the inductive learning algorithm is "perfect"—i.e., it identifies all dependencies that exist—this procedure will be guaranteed to work, because the independence assumptions will be correct. However, in practice, theories are often not perfect, due to limited data or an inadequate search heuristic. The result is that the procedure may not yield a valid distribution on G: the computed probabilities may sum to less than or more than one. In this case, we normalize the probabilities to sum to 1 and proceed as usual. In the extreme case, the sum of the probabilities will be zero if every goal outcome is assigned zero probability by some MSR. In this case, PCI assumes that not enough data has been collected to cover the current case adequately, and uses the less specific probability $P(G|f_*^s)$, where f_*^s is the set of features that are shared by all MSRs (possibly empty, in which case the prior probability $P(G)$ is used).

3.4.1 An Example of PCI

Taking the theory represented in Figure 3.4 as a predictive theory on a Boolean goal G, and leaving out the argument x, the theory can be rewritten as a set of conditional probabilities:

$$\begin{aligned} R_g: &\quad p_g = P(G) \\ R_a: &\quad p_a = P(G|A) \\ R_b: &\quad p_b = P(G|B) \\ R_c: &\quad p_c = P(G|C) \\ R_{ab}: &\quad p_{ab} = P(G|A \wedge B) \end{aligned}$$

In order to find any probability which is not explicitly represented in the theory, PCI must be applied. The simplest case is when only one rule applies to the new

probability. For example, for the situation D, R (the set of most specific rules) is just $\{R_g\}$, so

$$P(G|D) = P(G) = p_g$$

If the situation is $A \wedge B \wedge C$, R is $\{R_{ab}, R_c\}$. R_a and R_b are not in R, because R_{ab} applies and is more specific than either. Both rules are separable given the other, so either order is acceptable. The probability of G can then be computed using Equation 3.3:

$$P(G|A \wedge B \wedge C) = \frac{P(G|A \wedge B)\, P(G|C)}{P(G)}$$

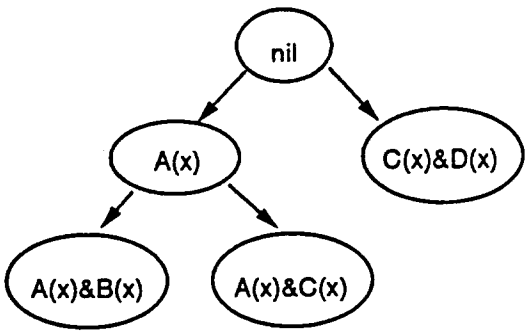

Figure 3.8: Theory to be used for making predictions

The theory in Figure 3.8 represents the probabilities p_g, p_a, p_{ab}, p_{ac}, and p_{cd}. If the situation is $A \wedge B \wedge C \wedge D$, R is $\{R_{ab}, R_{ac}, R_{cd}\}$. R_{ac} is not separable given R_{ab} and R_{cd}, since it shares the feature A with R_{ab} and C with R_{cd}. R_{ab} is separable given R_{ac} and R_{cd}, since it only shares features with R_{ac}, and R_{ac} is separable given R_{cd}, so a valid ordering is $R = (R_{ab}, R_{ac}, R_{cd})$. Applying Equation 3.3 gives

$$P(G|A \wedge B \wedge C \wedge D) = \frac{P(G|A \wedge B)\, P(G|A \wedge C)\, P(G|C \wedge D)}{P(G|A)\, P(G|C)}$$

$P(G|C)$ must be computed recursively: in this case, R is $\{R_g\}$, so $P(G|C) = P(G)$ and

$$P(G|A \wedge B \wedge C \wedge D) = \frac{P(G|A \wedge B)\, P(G|A \wedge C)\, P(G|C \wedge D)}{P(G|A)\, P(G)}$$

3.5 Planning

The planner in PAGODA performs a random action a fixed percentage of the time (default probability .25). The remainder of the time, it uses heuristic search to find the action with maximum overall expected utility. This overall expected utility is equal to the immediate expected utility of performing the action, plus the maximum utility of the plan that can be formed in the resulting states.

The planner forward chains to a fixed depth (default 3) through the space of possible outcomes for each action, then propagates the maximum expected utilities backwards to yield an expected utility for each initial action. This process can be described as an average-max search: at each level, the utility of the action with the highest expected (average) utility is used as the value to propagate back.

An example of part of the planning process to depth 2 is shown in Figure 3.9. Only the left half of the plan is fully expanded. Two actions, `:move-forward` and `:munch`, are considered. The theory used is as follows:

$$
\begin{aligned}
\Box \quad &\rightarrow_{.7} \quad \Delta u(t+1,-10) \\
&\rightarrow_{.3} \quad \Delta u(t+1,-11) \\
\texttt{action}(t, \texttt{:munch}) \quad &\rightarrow_{.5} \quad \Delta u(t+1,90) \\
&\rightarrow_{.5} \quad \Delta u(t+1,-10) \\
\texttt{action}(t, \texttt{:munch}) \wedge \Delta u(t,90) \quad &\rightarrow_{.67} \quad \Delta u(t+1,90) \\
&\rightarrow_{.33} \quad \Delta u(t+1,-10)
\end{aligned}
$$

The squares in Figure 3.9 represent predicted changes in utility; the capsules contain the expected utility of the entire plan below the capsule. The expected utilities in the bottom row are computed directly, using the probabilities to weight the predicted utilities. For example, the leftmost expected utility is equal to $(.67 * 90 + .33 * (-10))$. The values in the upper row of capsules are computed by taking the maximum expected utility of the rest of the plan from each state, plus the immediate utility of that state, and weighting by the probability of the state. For example, `:munch` has the highest expected utility in both of the lower states, so the expected utility of performing `:munch` (57 and 40 respectively for the two possible outcomes) is propagated backwards. The overall expected utility of performing `:munch` as the first

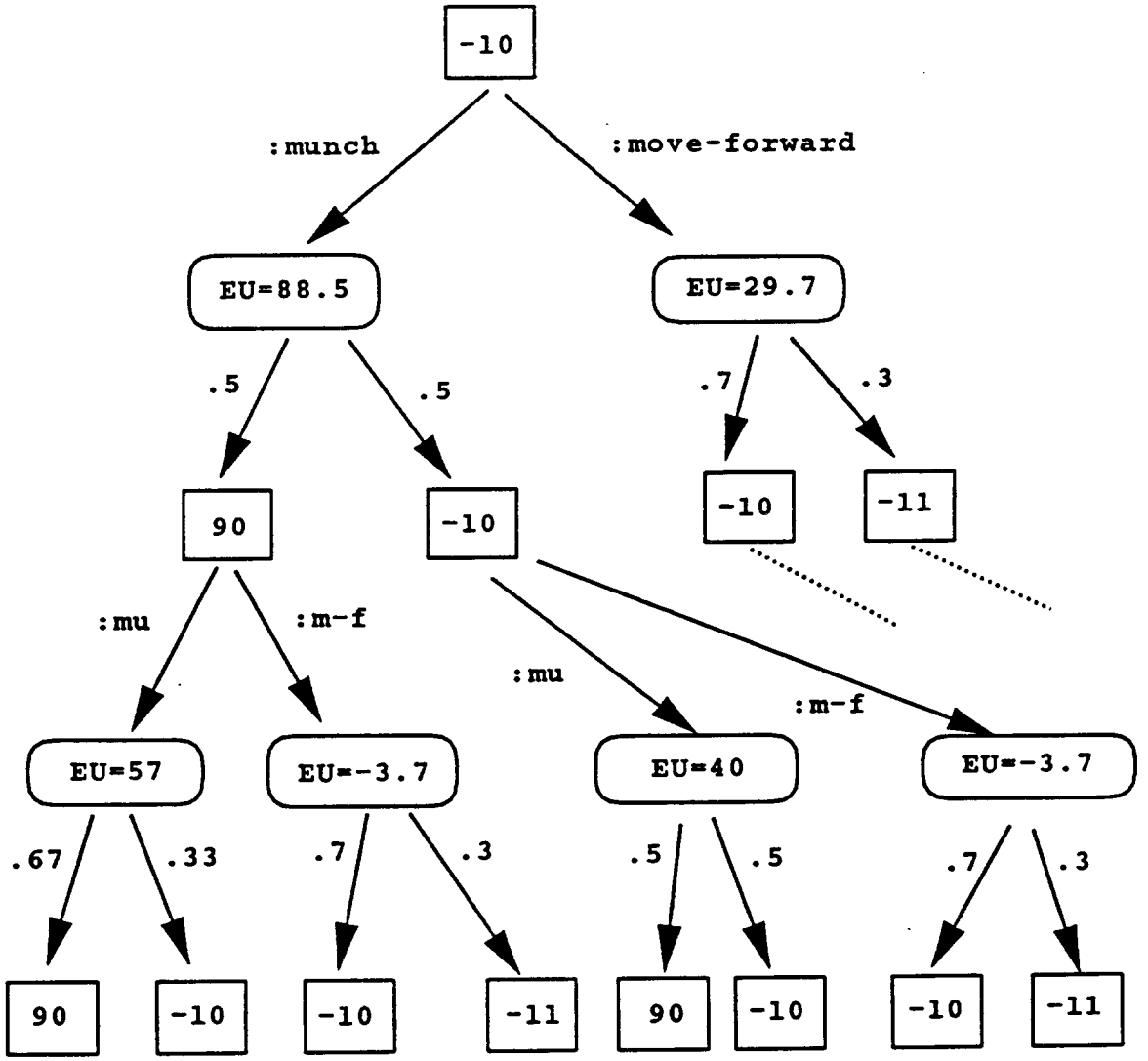

Figure 3.9: Partial plan search tree

action (represented by the upper left capsule) is then $(90+57)*.5+(-10+40)*.5$, or 88.5. The overall expected utility of `:move-forward` is 29.7 (computation not shown), so the planner selects `:munch`.

PAGODA's planning capacities could be enhanced by controlling and guiding the search process, and by integrating the planner more closely with the learning process. For example, the depth of the search might depend on the time available, the theory being used (e.g., if the agent has low confidence in the theory, it is not worth projecting very far ahead), and the plan computed so far (e.g., paths with low expected utility are less likely to be worth exploring). Also, the degree of exploration, and what actions the agent chooses when exploring, could be determined by the confidence the agent has in its theory, and by what aspects of the theory need refinement. These and other areas for future research are discussed in Chapter 9.

Chapter 4

Goal-Directed Learning

In order for an agent to function without a teacher, it must be able to select and classify its own learning examples. The agent will typically receive a large amount of sensory information from the environment, which (in our learning model) it must use to build a predictive model of the world. This model will consist of many different individual concepts (each feature of the world being predicted is a concept in the traditional sense). In complex domains, the environment will contain too many features for an agent with limited resources to learn in a reasonable amount of time. Therefore, it will need to focus its attention on aspects of the environment that are most relevant to its ability to succeed at whatever task it was built for: that is, it must decide what concepts to learn.

We have developed a theory called Goal-Directed Learning (GDL) that uses the principle of decision theory to choose learning tasks. The expected utility of being able to predict various features of the environment is computed and those with highest expected utility are used as learning goals.

> **Definition:** A learning goal is a feature of the world which the agent's inductive mechanism builds a model to predict.

For each learning goal, the learner uses background knowledge to select a learning bias (Chapter 5) and induces a predictive theory for the goal from observations of the world (Chapter 6).

An autonomous agent's primary task is to maximize expected utility. PAGODA does this by using a model of the world to make predictions about the effects of its actions on the world. Therefore, we provide PAGODA with utility as a primary learning goal, so that its initial theories predict the utility of actions in various world states.

As learning proceeds, the agent uses its existing theories to determine which features of the world it expects to be most useful to learn next. A feature is useful to learn if the plans formed by the agent lead to higher expected utility when the feature can be predicted than when it cannot be predicted. The most useful features are selected by the planner as learning goals.

Intuitively, the agent needs to be able to predict intermediate states in order to form plans to maximize utility in the long run. These intermediate states are learning goals, but are not necessarily planning goals (i.e., states the planner wants to achieve). For example, if the agent's utility is determined by the amount of money it has, and it has learned that putting a card into an ATM raises its utility, the next logical step is to learn how to get to an ATM. "Being at the ATM" would then be formed as a learning goal. On the other hand, if PR learns that standing next to a nasty frequently leads to a large loss in utility, then being able to predict this state is useful, so that plans can be formed to avoid it. "Being next to a nasty" would be a useful learning goal, but not a planning goal.

In the simple world shown in Figure 4.1, PR may learn the following theory about utility:

$$\begin{aligned} &\texttt{food-smell}(t, 20) \wedge \texttt{action}(t, \texttt{:munch}) \rightarrow_{1.0} \Delta\mathrm{u}(t+1, 90) \\ &\Box \rightarrow_{1.0} \Delta\mathrm{u}(t+1, -10) \end{aligned} \tag{4.1}$$

$\texttt{food-smell}(t, 20)$ is true in this world if and only if PR is standing on food. If this is not the case, the agent believes that all actions have equal utility, and will wander randomly until it happens to land on food. At that point, it recognizes that :munch is the best action, and proceeds to eat the food. However, if it could predict which actions and states lead to $\texttt{food-smell}(t, 20)$, it would be able to plan ahead—any time it was next to and facing food, the best two-step sequence would be (:move-forward,

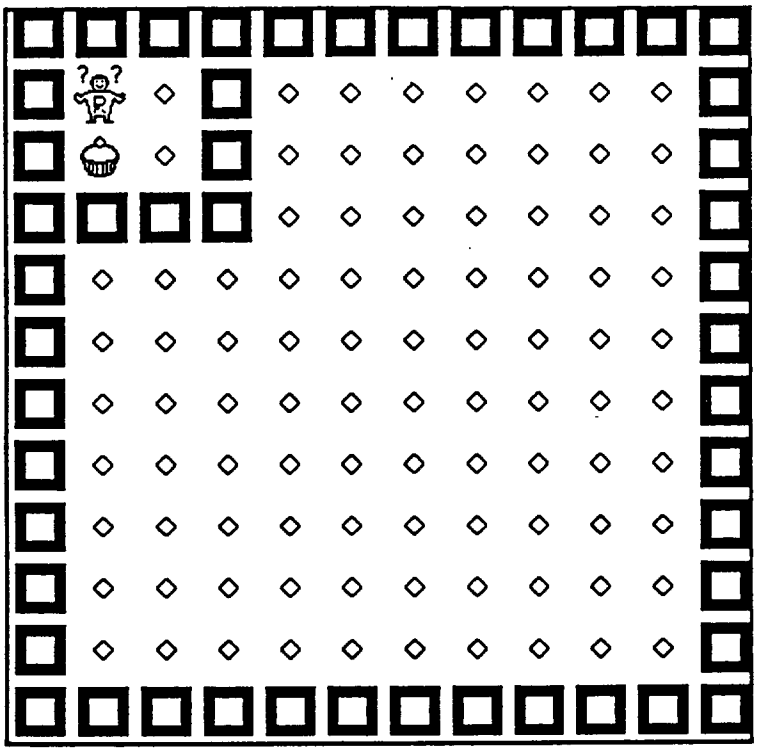

Figure 4.1: A simple RALPH world

:munch).

In Section 4.1 we derive the value of information formula used to evaluate the effect of a proposed learning goal on utility; Section 4.2 explains how the expected utility of plans using a specified learning goal is computed. Section 4.3 gives an example of the application of GDL to a theory in the RALPH domain.

4.1 Utility of Learning Goals

For an optimal (unbounded) rational agent, the value of a learning goal F (a feature of the environment) is the agent's expected utility per unit time given a world model which predicts F in addition to the set of current learning goals, L, minus the expected utility per unit time given the current world model, which only predicts L. If P is the planning function for the agent, taking the knowledge available in the world model (i.e., the set of learning goals) as its argument and returning the

expected utility of a plan formed using that knowledge, then the value of a learning goal F for an optimal agent is:

$$V_{\text{opt}}(F) = P(L \wedge F) - P(L) \tag{4.2}$$

For a limited rational agent, the value of a learning goal must take into account the cost of learning the goal. These costs include utility lost due to computation time expended during learning (instead of planning or acting), additional planning costs with the resulting, more complex, model, and costs of experimentation necessary to acquire sufficient data. The costs depend on the particular agent and the environment; we model them as a single cost function $C(F)$. The net utility of a learning goal F is

$$V_{\text{lim}}(F) = P(L \wedge F) - P(L) - C(F) \tag{4.3}$$

A limited rational agent should be willing to learn goals that have positive net utility, i.e., for which $V_{\text{lim}}(F) > 0$, and should be indifferent to learning goals with zero net utility. Of course, the costs and benefits of learning multiple goals may not have a simple additive effect. For example, in a medical domain, learning whether a patient has stomach cancer or an ulcer may both be useful initially, but once the patient is known to have cancer, learning whether an ulcer is also present may become less useful. The exact change in predictive value depends on the decision maker's state of knowledge about the domain and the particular situation of interest.

Additionally, the time spent learning multiple goals may exceed the available amount of time in a real time environment; if the agent has "anytime" learning algorithms that be run for a shorter period of time, the costs may be manageable, but the accuracy of the learned model—and therefore the expected utility of planning with the model—will decrease. An ideal agent should learn the set of goals that, taken together, maximize its expected utility.

In PAGODA, the cost of learning is ignored when computing the value of learning goals, so the value of a goal is based on the formula for an optimal, unbounded agent (Equation 4.2).

Definition: The single-step model treats the difference in two plans

generated using different world models as the difference in the final step of each plan.

The final step of each plan is the action selected by the agent when it has chosen (by constructing a plan) values for all of the learning goals predicted by the world model. Two assumptions underlie the single-step model: first, the utility of the intermediate steps in the plans is assumed to have the same expected value. In fact, the utility of these intermediate steps in the plans will vary, but since the new model has not yet been learned, the agent cannot determine the difference in the utility of the intermediate steps.

Second, the agent is assumed to be able to control the values for its learning goals. That is, it is assumed to be able to achieve states in which its learning goals take on whatever values the agent chooses. In fact, since there is uncertainty in its world model, it can only predict their values probabilistically. Additionally, some features will be more controllable than others. For example, the agent can control its location to a greater extent than it can control the weather. However, since our model assumes no *a priori* knowledge of the degree of control the agent can exercise over various features, the expected degree of control (from the agent's point of view) is the same for all features.

Under the single-step model, the agent generates a plan to achieve a state of the world in which the values of its learning goals maximize its expected utility (given the remaining features of the world, which it cannot control), and then takes the action which maximizes its utility in that state. This final action is the *single-step plan* chosen by the agent. Therefore, applying the single-step model to Equation 4.2 gives the following value of a goal in PAGODA:

$$V(F) = P_1(L \wedge Z) - P_1(L) \tag{4.4}$$

where P_1 is a planning function that returns the expected utility of the single-step plan, given that the planner can choose values for the specified learning goals.

In the next section, we derive the expected utility of this single-step plan, given an arbitrary set of learning goals Z. The conditioning contexts of rules are

assumed to consist of conjunctions of feature values (although internal disjunctions—i.e., multiple values for a given feature—are allowed). The representation used in PAGODA, which was described in Chapter 3, satisfies this assumption. The derivation is also based on the assumptions discussed above. Extensions that would allow some of these assumptions to be relaxed or modified are discussed in Chapter 9.

4.2 Utility of Plans

A single-step plan is defined to be a triple (a, pw, u), where a is an action, pw is a perceived world, and u is the expected utility of taking a in pw. The set of actions and perceived worlds are determined by the environment; the expected utilities are computed from the agent's utility theory. Each rule in the theory corresponds to the set of single-step plans which have an action and perceived world for which the rule is the most specific rule in the theory. The expected utility associated with each of these single-step plans is the expected utility of the rule.

The past frequency of application of a set of single-step plans corresponding to a particular rule is equal to the number of past instances covered by the rule:

> **Definition:** The **past instances** of a rule are the observations that were covered by the rule (i.e., used to make predictions).

The average utility per unit time of the past instances can be found by weighting the expected utility of each rule by its relative past frequency (i.e., its number of past instances, divided by the total number of past instances). In order to determine the expected utility per unit time of future plans, we construct a **hypothetical theory**. The rules in the hypothetical theory have the same conditioning contexts and expected utilities as the rules in the current utility theory, but are weighted by the rule's hypothetical instances, rather than its past instances:

> **Definition:** The **hypothetical instances** of a rule are the observations that would have been covered by the rule if the agent could have chosen the values for the learning goals Z in order to maximize its expected utility.

Under the single-step model, given a set of single-step plans (a_i, pw_j, u_{ij}), where the pw_is vary only in the values they specify for the learning goals, the agent can choose which state pw_j to be in and which action a_i to take, in order to maximize its expected utility u_{ij}. The probability with which a rule (set of single-step plans) is chosen is given by its relative hypothetical frequency (the rule's number of hypothetical instances divided by the total number of instances).

We let n_r represents the number of past instances for a rule r; m_r is the number of its hypothetical instances.

For each rule r, if there are rules S_r that can be substituted for r (by choosing values for the action and for the learning goals Z) and have higher utility than r, the past instances of r will be included in the hypothetical instances of the rule in S_r with highest utility. When all rules have been examined, the hypothetical instances are used to compute the expected utility of the resulting hypothetical theory.

$\mathcal{R}$:= { r : n[r] $\neq$ 0 }
foreach r in $\mathcal{R}$:
    m[r] := 0
foreach r in $\mathcal{R}$:
    Substitution-Set (r) := {r} $\cup$
        { e in $\mathcal{R}$: ($\bar{Z}$[r] $\rightarrow$ $\bar{Z}$[e]) $\wedge$ $\neg$ ((Z[e] $\equiv$ Z[r]) $\wedge$ (action[e] $\equiv$ action[r]))
    u := argmax[s in Substitution-Set(r)] Expected-Utility (s)
    m[u] := m[u] + n[r]
N := sum[r in $\mathcal{R}$] (m[r])
EU (K (Z)) := (1 / N) * sum[r in $\mathcal{R}$] (m[r] * Expected-Utility (r))

Figure 4.2: Algorithm for finding the expected utility of a plan.

The algorithm for finding the expected utility of a plan is given in pseudo-code in Figure 4.2. First, those rules which have never actually been applied in the past (i.e., for which $n_r = 0$) are removed from the theory. These rules represent generalizations which may be used by the inference algorithm, but are not relevant for this algorithm. The remaining rules are initialized to have zero hypothetical

instances.

> **Definition:** The substitution set for r is the set of all rules that could be substituted for r by changing the action chosen and/or the values of the learning goals Z.

Letting $\bar{Z}$ stand for the features in the world that are not included in the learning goals Z, an alternative rule e can be substituted for r whenever (1) e's $\bar{Z}$ feature values are implied by those of r, and (2) e's Z feature values or action differ from those of r. If (1) does not hold, the set of action/perceived world pairs corresponding to e will not include those of r. If (2) does not hold, the agent would not be able to substitute e for r by changing the values of the action and learning goals. Every rule is also a member of its own substitution set. The past instances of r are assigned to the hypothetical instances of the rule in r's substitution set with the highest utility (which could be r itself).

After all of the substitution sets have been computed, and hypothetical instances determined, the overall expected utility is computed by weighting the expected utility of each rule r with m_r, its hypothetical frequency of application.

4.3 An Example in the RALPH World

Figure 4.1 shows a simple RALPH world, containing only PR and one piece of food. Whenever PR consumes the food, a new piece appears randomly. PAGODA learns the following theory to predict its change in utility in this world:

$$
\begin{array}{ll}
R_1(n_1 = 60) & \Box \rightarrow_{1.0} \Delta \mathrm{u}(t+1, -10) \\
R_2(n_2 = 6) & \texttt{action}(t, \texttt{:move-forward}) \wedge \texttt{vision}(t, \texttt{wall}, 1) \rightarrow_{1.0} \Delta \mathrm{u}(t+1, -11) \\
R_3(n_3 = 9): & \texttt{food-smell}(t, 20) \wedge \texttt{action}(t, \texttt{:munch}) \\
 & \qquad \rightarrow_{1.0} \Delta \mathrm{u}(t+1, 90)
\end{array}
$$

The set of learning goals Z for this world model is $\{\Delta u\}$. The substitution sets for the three rules in the theory are:

$$\begin{aligned} S_1 &= \{R_1\} \\ S_2 &= \{R_2, R_1\} \\ S_3 &= \{R_3, R_1\} \end{aligned}$$

R_2's substitution set is $\{R_2, R_1\}$ because $\texttt{vision}(t, \texttt{wall}, 1)$ $(\bar{Z}[R_2])$ implies $\Box$ $(\bar{Z}[R_1])$ and $\texttt{action}(t, \texttt{:move-forward})$ $(\texttt{action}[R_2])$ differs from $\Box$ $(\texttt{action}[R_1])$. Since R_1 has higher expected utility than R_2, R_2's past instances (6 observations) are included in R_1's hypothetical instances. R_3 has higher expected utility than R_1, so its past instances are included in its hypothetical instances; R_1 is the only rule in its substitution set, so its past instances are included in its hypothetical instances. The final hypothetical instance counts are:

$$\begin{aligned} m_1 &= 66 \\ m_2 &= 0 \\ m_3 &= 9 \end{aligned}$$

The expected utility of this hypothetical theory is the weighted sum of outcomes:

$$\frac{1}{75} * (66 * (-10) + 0 * (-11) + 9 * 90) = 9.92$$

To compute the expected utility of the learning goal **vision**, we first find the hypothetical theory when $Z = \{\texttt{vision}, \Delta u\}$, then compute the expected utility of this hypothetical theory, and subtract the expected utility of the initial theory.

The substitution sets S_r for the three rules are as follows:

$$\begin{aligned} S_1 &= \{R_1, R_2\} \\ S_2 &= \{R_2, R_1\} \\ S_3 &= \{R_3, R_1\} \end{aligned}$$

R_2 can be substituted for R_1 because their left-hand sides differ only in features in Z and the action; R_3 cannot be substituted for R_1 because $\texttt{food-smell}(t, 20)$ is not implied by $\Box$. R_1 can be substituted for both R_2 and R_3 because its Z features

(empty) are different from R_2 and R_3, and its Z features (also empty) are implied by both.

Within these three substitution sets, the rules with the highest expected utility are R_1, R_1, and R_3, respectively. Therefore, R_2's 6 past instances are assigned to the hypothetical instances of R_1, resulting in the following final weights:

$$\begin{aligned} m_1 &= 66 \\ m_2 &= 0 \\ m_3 &= 9 \end{aligned}$$

The expected utility of the resulting hypothetical theory is

$$\frac{1}{75} * (66 * (-10) + 9 * 90) = 9.92$$

The value of the candidate learning goal `vision` is therefore 9.92 - 9.92, or 0. Notice that although R_1's substitution set is different from that in the original theory, the final weights are the same, and therefore `vision` has no value as a learning goal. Intuitively, learning about `vision` doesn't gain PR anything in this theory: the agent already knows enough not to bump into the wall, and predicting `vision` only allows it to avoid situations where it could bump into the wall if it didn't know better.

The expected utility of `food-smell` is computed similarly. The substitution sets in this case are as follows:

$$\begin{aligned} S_1 &= \{R_1, R_3\} \\ S_2 &= \{R_2, R_1\} \\ S_3 &= \{R_3, R_1\} \end{aligned}$$

The rules with the highest expected utility are, respectively, R_3, R_1, and R_3. R_1's past instances are therefore assigned to R_3's hypothetical instances and R_2's past instances are assigned to R_1. The resulting final weights are:

$$\begin{aligned} m_1 &= 6 \\ m_2 &= 0 \\ m_3 &= 69 \end{aligned}$$

Note that the the re-assignment of instances is not transitive: R_2's past instances cannot be moved to R_3, because R_3 does not specify `food-smell` (which is in $\bar{Z}$ and therefore cannot be changed).

The expected utility of the resulting hypothetical theory is

$$\frac{1}{75} * (6 * (-10) + 69 * 90) = 82$$

The value of the candidate learning goal `food-smell` is 82 - 9.92, or 72.08.

`food-smell` is clearly a better learning goal than `vision`. Predicting it allows the agent to plan to be at food, where it can eat the food and raise its utility.

Intelligent agents must determine what properties of the world are most worth learning about if they are to behave autonomously and rationally. GDL provides a theoretically justified technique, based on decision-theoretic analysis, for determine the value of learning features of the world. Both a general model for the value of learning goals in a rational agent, and a particular model for PAGODA's performance element, are given by GDL. The primary open question is how the agent can determine not just the benefit of a learning goal, but its net utility, including the costs of learning.

Chapter 5

Selecting a Learning Bias

Selecting a hypothesis language for an intelligent learning system—whether this is done by the designer or by the learning system itself—is in effect stating a set of beliefs about how the world is expected to work. In defining the form of the theories, their syntactic and semantic content, and even the primitives from which they may be constructed, the scope of the agent's understanding of the world is constrained.

The probabilistic evaluation method described in Chapter 6 uses the simplicity of theories to define their prior probability. The underlying assumption is that the language used is appropriate for the theories to be learned, in the sense that the agent expects an efficient model of the world to be describable using the language. This approach reflects a belief that representation languages should be chosen so that the agent's fundamental beliefs about the structure of the world are embedded in the language.

The idea that language reflects one's knowledge about the world, and that in an ideal language, truth would be easily expressible, is not a new one. Long ago there was believed to be an *Urlanguage*, spoken by Adam, in use until the fall of Babel, which was in complete harmony with the universe: "writ on the firmament.... inscribed by signature on every leaf and stone." ([Hacking, 1975], p. 80)

Later, science supplanted religion, but the concept of an ideal language was not lost. Leibniz, for example,

> ...did not believe in lost innocence but rather in a science and a language

> that more and more closely correspond to the structure of the universe. ([Hacking, 1975], p. 140)

Still more recent work such as Goodman's treatise on induction and the projectibility of predicates [Goodman, 1955] and Rosch's research on the psychology of basic concepts (see, for example, [Rosch *et al.*, 1976] and [Rosch and Lloyd, 1978]) emphasize the fact that we embed knowledge in our language: in a useful language, frequently used (and useful) concepts can be expressed easily. Language evolves to reflect our beliefs about what is likely to be true. For example, simple concepts—those which can be expressed succinctly in our language—are psychologically preferred by people, all other things being equal. A language defines an informed prior probability distribution, in the sense that it incorporates the knowledge used to create the language.

On the other hand, when no "good" hypothesis can be found in a language, the language is not expressive enough, and a better language may need to be found. Similarly, complex hypotheses that are found to have a high posterior probability may indicate that new terms should be introduced to make these theories simpler to express ("quark" and "gravitational force" are two examples of new terms in the history of science that made complex theories easier to express).

Since PAGODA defines the prior probability of theories in terms of simplicity, a function of the language used, what does it mean to change the language (and thus change the prior)? It seems somewhat paradoxical to change the "prior" (as defined by the language) at all, and in particular, to use posterior probabilities—computed using the prior as a starting point—to modify the prior. However, as we mentioned above, the probability distribution defined by the language is an *informed* prior, incorporating whatever information was used in the decision to change the language. Updating the language is then equivalent to computing a better prior which can be used in the next iteration.

In this chapter, we describe Probabilistic Bias Evaluation (PBE), a method for using probabilistic background knowledge to select maximally relevant features to describe theories, thus lowering the computational complexity of the learning task. Smaller theory spaces are easier and faster to search (i.e., it takes fewer examples to

converge on the best hypothesis in the space, and the amount of time spent processing each observation is lower), enabling an agent to focus its limited computational resources. On the other hand, larger spaces are more likely to contain good hypotheses. We present a formal analysis of this tradeoff, and show that a value can be assigned to each potential language bias.

Traditional empirical learning systems use learning biases provided directly by the designer to generate theories to describe data. Explanation-based learning approaches, on the other hand, require the designer to provide a complete domain theory, which is used to generalize single observations into operational theories. PBE bridges the gap between empirical approaches and explanation-based learning: the learning bias is derived automatically from background knowledge consisting of a partial, probabilistic domain theory.

In the next section, related work on bias is discussed briefly. Section 5.2 presents the motivation for the probabilistic bias evaluation technique used in PAGODA; Sections 5.3 to 5.6 build the formal theory. In Section 5.7 we give the formula for the value of a bias and present a closed-form solution for the RALPH world. Finally, results demonstrating the need for evaluating bias are given in Section 5.8. These results use ID* (Appendix B), an incremental decision tree learning algorithm based on [Quinlan, 1986] and [Utgoff, 1988], to show the effects of selecting various biases on a traditional learning algorithm. The empirical tests described in Chapter 7 show the effects of bias selection in PAGODA.

5.1 Background

Bias refers to a restriction on or preference within the space of theories considered by a learning system. Without some bias, there would be no basis for making inductive generalizations [Mitchell, 1980]. A strong, correct bias (i.e., one that permits only likely theories) is extremely useful to have because it allows a learner to converge quickly to a good theory. How to find a good learning bias has been an open research question; in this chapter, we describe Probabilistic Bias Evaluation (PBE), a method for using probabilistic domain knowledge to assign values to potential biases

for each learning goal. PAGODA uses the best of the evaluated biases to constrain the hypothesis space for the learning goal. The preference function within the constrained space is given by one of the prior probability distributions discussed in Chapter 6.

Russell and Grosof [1987] showed that bias can be represented declaratively for deterministic learning problems as background knowledge in the form of determinations. In probabilistic domains, choosing a bias is a more complex process. The bias evaluation technique described in this chapter uses probabilistic background knowledge to evaluate biases.

The Minimum Length Encoding (MLE) principle states that the total length of a theory and the data encoded using the theory should be minimized. MLE embodies a bias towards simple theories, traded off with accuracy, since an inaccurate theory will not encode the data efficiently. PAGODA uses a similar bias in its preference for simple theories; this preference is part of the Bayesian theory evaluation technique described in Chapter 6.

The next three sections give some more background on bias, declarative bias, and MLE techniques.

5.1.1 Bias in Machine Learning

The hypothesis language used by a learning system defines what it can learn and imposes a particular structure on its learned theories. A particular theory may be easily and efficiently describable in one language, require cumbersome definitions in a second language, and be inexpressible in a third. The bias that constrains the language may come from a variety of sources, including prior beliefs, syntactic preferences, the sensory inputs available, and the agent's vocabulary.

In PAGODA, the bias derives from three sources: PBE is used to select a set of features to describe hypotheses; each hypothesis is required to be a Uniquely Predictive Theory (Chapter 3); and a simplicity metric is used to define the prior probability distribution (Chapter 6).

The ability to *change* bias is as important as initial bias selection for autonomous agents. In order to operate in unanticipated environments, they cannot

depend on the programmer to build in the "correct" (i.e., most efficient and accurate) language, so they must be able to change their representation. This means that any assumptions made by the designer about what language is most likely to be useful must be made explicitly, and the agent must be able to override these design decisions. In the current design of PAGODA, bias is fixed when learning commences. Techniques for shifting bias during the learning process are proposed in Chapter 9.

5.1.2 Declarative Bias

A determination, as defined in [Davies and Russell, 1987], represents a dependency between relational schemata. P determines Q ($P \succ Q$) if all objects that share the same P-value also share the same Q-value. Formally,

$$P(x,y) \succ Q(x,z) \quad \text{iff} \quad \forall wyz[P(w,y) \wedge Q(w,z) \rightarrow \forall x[P(x,y) \rightarrow Q(x,z)]] \tag{5.1}$$

Background knowledge in the form of determinations can be used to derive bias [Russell and Grosof, 1987]. Suppose an agent wants to learn how to predict change in utility, $\Delta \mathrm{u}(t,u)$, and the background knowledge consists of a single determination:

$$\texttt{vision}(t,o,d) \wedge \texttt{action}(t,a) \succ \Delta \mathrm{u}(t+1,u) \tag{5.2}$$

The arguments to `vision` are an object, o, and its distance, d. If the agent sees an instance of the form

$$\texttt{vision}(5,\texttt{food},0) \wedge \texttt{action}(5,:\texttt{eat}) \wedge \Delta \mathrm{u}(6,100)$$

it can conclude the rule

$$\texttt{vision}(t,\texttt{food},0) \wedge \texttt{action}(t,:\texttt{eat}) \Rightarrow \Delta \mathrm{u}(t+1,100)$$

We can think of the determination in Equation 5.2 as a rule template of the form

$$\forall t\, \texttt{vision}(t,o,d) \wedge \ \texttt{action}(t,a) \rightarrow \Delta \mathrm{u}(t+1,u) \tag{5.3}$$

Notice that t is universally quantified but o, d, a, and u are free. This is because of their asymmetric role in the definition of determinations: t functions as the generic object x whose Q-value z is determined by the P-value y in Equation 5.1. In the final rule, t is universally quantified; e, a, and u may be instantiated to a specific or generalized value, or left uninstantiated, in which case they are considered to be universally quantified. For example, if a is uninstantiated, all actions in the specified state of the environment have the same outcome.

In general, we can use multiple determinations, chaining them together, and take advantage of the derivation process to get a *tree-structured bias* [Russell, 1988], in which the interactions between the predicates in the theory are constrained by the structure of the derivation. A method of finding a maximally operational tree-structured bias is outlined in [Getoor, 1989].

Declarative bias using determinations is straightforward because the bias is expressed as a first-order logical sentence, and learning consists of applying deductive logic to the bias and observations to yield consistent rules which can then be further generalized using inductive techniques. However, determinations are rarely available in uncertain environments, where theories to be learned may be nondeterministic. PBE provides a method for using probabilistic background knowledge to impose a bias on learning.

5.1.3 Minimum Length Encoding

The Minimum Length Encoding (MLE) principle states that the best theory is the one that minimizes the length of coding the theory and the observations with respect to the theory. (See Chapter 2 for an introduction to MLE.) This length is given by the length of the shortest input to a Universal Turing Machine (UTM) that causes the UTM to generate the observations as output. The coded theory is the part of the input that causes the UTM to simulate some target machine; the coded data is the part of the input that is treated as input to the (simulated) target machine, which causes that machine to generate the observations as output. One can prove that applying the MLE principle using *any* UTM to encode the target machine and

data will converge to the correct theory, given enough data.

In practice, of course, we don't have a UTM available, can't afford to use it anyway, and are more interested in relatively immediate results than in convergence proofs. Practical applications of the MLE principle (e.g., [Pednault, 1989] and [Babcock *et al.*, 1990]) use hand-tailored coding schemes for the application at hand (which are usually not equivalent to any UTM because they implicitly limit the language used). Presumably, the designer chooses a language which will be reasonably close to the optimal language for the problem. However, if this language is not the most efficient, the application may converge very slowly; if the language is inadequate, the correct theory will not be found at all.

In PAGODA, instead of providing a language, the designer provides some general background knowledge which the agent uses to derive a good representation dynamically. Because the representation can be changed, and the background knowledge can be modified,[1] an error on the designer's part is not fatal to the system's performance.

5.2 Probabilistic Evaluation of Bias

The goal of PBE is to define a quantitative criterion for evaluating biases, allowing an agent to find the "best" bias (with respect to this criterion) for learning to predict a specified learning goal. For the rest of this chapter, "bias" will refer to the set of features used to make predictions, unless otherwise specified.

The best bias will not be the one which is most likely to contain the "correct" theory (assuming there is such a theory), since any superset of a given bias will always be better—or at least as good—by this measure. Instead, the criterion should trade off accuracy (in the sense of making good predictions) with simplicity, so that a feature that would significantly increase the size of the hypothesis space to be searched while only yielding slightly better theories will not be considered adequately relevant, and will not be included in the bias. Specifically, the best bias in PBE is the one which

[1]This is true in principle; we do not actually learn the background knowledge in this work.

maximizes the agent's predictive accuracy over time, subject to any preference it may have for short-term as opposed to long-term performance.

The consequences of selecting a larger theory space are twofold. First, it will take more observations to converge on a good theory. In discrete embedded environments such as RALPH, at most one relevant observation can be collected per time step (an "observation" in this sense is simply a sensory experience). Therefore, using a larger theory space, more time will pass in the real world before the agent begins to make good predictions. The degree to which this time matters depends on how much the agent discounts future performance.

Second, searching a larger space takes more computational time *per observation.* If the agent has limited computational power and has other tasks pending—which is presumably the case in a complex environment—learning a concept using a large theory space may interfere with its ability to perform these other tasks. In cases where the space is very large, the agent may not even be able to keep up with the stream of observations provided by the environment. The effect of this depends on the cost of time in the environment.

In the remainder of this chapter, *time* should be taken to refer to the number of observations made, unless otherwise indicated. However, PBE trades off the number of observations against expected accuracy, which will tend to favor simpler biases (since these require fewer observations to converge to a good theory). Therefore, the resulting biases will also tend to minimize computational time spent per observation, since the search space, and the theories themselves, will be smaller.

The value that PBE assigns to each potential bias is a measure of that bias's expected behavior in the long run. The accuracy of the agent's predictions over time will depend on the bias selected: a small feature set with high predictiveness (i.e., a high uniformity) will converge quickly to high accuracy. Conversely, a large feature set with low predictiveness will take a long time to converge to relatively poor accuracy. The choice between two such biases is clear; the real question is what the agent should do when the choice is between two biases when one is larger but also more predictive. The decision will depend on how strongly the agent prefers short-term performance to long-term performance.

To model this tradeoff, we define several concepts:

> **Definition:** The **expected accuracy** of a bias is the accuracy of predictions that the agent believes the best (i.e., most accurate) theory in the bias will make.

> **Definition:** An agent's **learning curve** for a particular bias and learning goal is the expected accuracy of predictions on the learning goal using the specified bias, given as a function of time.

The general shape of a learning curve depends on the agent's learning algorithm; the particular shape and asymptote depend on the syntactic and semantic properties (including the expected accuracy) of the bias.

> **Definition:** An agent's **time-preference function** is a measure of how much a reward at a particular time is worth to the agent.

For example, a flat time-preference function is equivalent to having no preference for short-term or long-term reward.

To find the value of a bias, we first determine its expected accuracy, using background knowledge provided by the designer. Next, the learning curve for the bias is determined, based on the expected accuracy. Finally, the learning curve is combined with the time-preference function, yielding an overall expected accuracy, weighted by the agent's time preferences. Because the time-preference function used in PAGODA is a discounting function (under which future performance is discounted, or devalued, proportional to its distance in time), this overall accuracy is referred to as the **expected discounted future accuracy**. This is the value of the bias; whichever proposed bias has the highest value is used for learning.

5.3 Probabilistic Background Knowledge

Determinations (Section 5.1.2) are *weak* knowledge, in that they do not specify what the function mapping the inputs P to the outcome Q is, only that one exists. So, for example, if we know that $\text{Species}(x, s) \succ \text{Color}(x, c)$, we won't be able to predict the color of an individual of a previously unobserved species. But

if we have observed one individual of species s_1 whose color is known to be c_1, we can immediately form a rule that says $\forall x[\text{Species}(x, s_1) \rightarrow \text{Color}(x, c_1)]$. We refer to the latter sort of rules—which enable individual predictions to be made—as *strong* knowledge. Weak knowledge in the form of determinations can be used to determine which features $\mathcal{F}$ are relevant for predicting $\mathcal{O}$ and thus extract a representation for learning strong theories to predict $\mathcal{O}$.

Probabilistic background knowledge about relevance is represented in PBE using a form of weak knowledge called *uniformities*, which are a probabilistic version of determinations. Uniformities are similar to partial determinations as defined in [Russell, 1986]. $U(\mathcal{O}|\mathcal{F})$ (read "the uniformity of $\mathcal{O}$ given $\mathcal{F}$") is the probability that two individuals sharing the same $\mathcal{F}$-value will share the same $\mathcal{O}$-value. Roughly, it is the degree to which $\mathcal{O}$ can be predicted given $\mathcal{F}$. As with determinations, it does not specify what the most common $\mathcal{O}$-value will be for any given $\mathcal{F}$-value. Formally,

$$U(\mathcal{O}|\mathcal{F}) = P(\mathcal{O}(x) = \mathcal{O}(y)|\mathcal{F}(x) = \mathcal{F}(y)) \tag{5.4}$$

Initial uniformities will be provided by the system designer or by domain experts. Although this is not trivial, it is at least easier than providing the complete deterministic domain theories needed by traditional explanation-based learning systems. The initial uniformities can be updated as the system acquires experience. Techniques for learning uniformities are similar to those for learning strong theories [Russell, 1986].

Uniformities are similar to the variability bias described by [Martin and Billman, 1991]. Instead of just summarizing the distribution of outcomes, a variability bias specifies the expected distribution more precisely. The distribution gives the probabilities of the most common outcome (for an arbitrary value of the input features), the second most common outcome, and so on, again without specifying what those values are. Variability biases, which are based on studies of human learning and are intended to represent a form of knowledge that humans appear to use, may be a useful extension when more information is known about the shape of the outcome distribution.

5.4 Expected Accuracy

To find the expected accuracy of predictions over time, PBE first computes the expected accuracy of the best theory in the space defined by $\mathcal{F}$. Recall that the uniformity of $\mathcal{O}$ given $\mathcal{F}$ specifies the probability that two randomly selected individuals with the same $\mathcal{F}$-value will have the same $\mathcal{O}$-value. Using this uniformity value and making some assumptions about the distribution of outcomes, $\hat{p}$, the probability of the most likely outcome $\hat{o}$, is computed for an arbitrary value of $\mathcal{F}$. The best theory is the one that always predicts the most likely outcome in each situation; $\hat{p}$ gives this theory's expected accuracy.

We assume a simple prediction task: every time a new example ($\mathcal{F}$-value) is observed, the agent must predict a value for $\mathcal{O}$. If the most likely outcome is always predicted (maximizing expected accuracy), the expected accuracy of the best theory is $\hat{p}$, as given below.[2]

The distribution which maximizes entropy (Section 2.3.1) will be the one which is closest to a uniform distribution. In this case, the maximum entropy distribution satisfies the following assumptions (the proof is given in Appendix A):

1. For each value of $\mathcal{F}$, there is one $\mathcal{O}$-value, $\hat{o}$, which occurs most often.

2. The other values of $\mathcal{O}$ occur equally often.

Suppose that $\mathcal{O}(x)$ can take on n different values, $o_1, \ldots, o_n$. Assumption #1 says that given $\mathcal{F}$, some $\hat{o}$ has the highest probability. Without loss of generality, assume that this is o_1; its probability is $\hat{p}$:

$$P(\mathcal{O}_1|\mathcal{F}) = \hat{p} \tag{5.5}$$

where $\mathcal{O}_i$ stands for the event that $\mathcal{O}(x) = o_i$.

Assumption #2 says that the remaining $\mathcal{O}$-values have equal probability. If there are n values of $\mathcal{O}$,

$$P(\mathcal{O}_i|\mathcal{F}) = \frac{1-\hat{p}}{n-1}, \qquad i = 2, \ldots, n \tag{5.6}$$

[2]If outcomes are instead predicted according to their expected probability, and the assumptions about the distribution given below are made, the expected accuracy will be equal to the uniformity $u_{\mathcal{O}\mathcal{F}}$. This will always be less than or equal to $\hat{p}$.

Introducing a shorthand notation in Equation 5.4 gives:

$$U(\mathcal{O}|\mathcal{F}) = u_{\mathcal{O}\mathcal{F}} = P(\mathcal{O}(x) = \mathcal{O}(y)|\mathcal{F}(x) = \mathcal{F}(y))$$

For convenience of notation, we remove the conditioning context: for the remainder of this derivation, it is implicitly assumed that $\mathcal{F}(x) = \mathcal{F}(y)$. Substituting the probabilities from Equations 5.5 and 5.6, and solving for $\hat{p}$ in terms of $u_{\mathcal{O}\mathcal{F}}$ (remember that x and y are independent random variables):

$$\begin{aligned}
u_{\mathcal{O}\mathcal{F}} &= P(\mathcal{O}(x) = \mathcal{O}(y)) \\
&= \sum_{i=1}^{n} P(\mathcal{O}(x) = \mathcal{O}(y) = o_i) \\
&= P(\mathcal{O}(x) = \mathcal{O}(y) = o_1) + \sum_{i=2}^{n} P(\mathcal{O}(x) = \mathcal{O}(y) = o_i) \\
&= [P(\mathcal{O}_1)]^2 + \sum_{i=2}^{n} [P(\mathcal{O}_i)]^2 \\
&= \hat{p}^2 + (n-1)\left(\frac{1-\hat{p}}{n-1}\right)^2 \\
&= \hat{p}^2 + \frac{(1-\hat{p})^2}{n-1} \\
(n-1)u_{\mathcal{O}\mathcal{F}} &= (n-1)\hat{p}^2 + 1 - 2\hat{p} + \hat{p}^2 \\
&= n\hat{p}^2 - 2\hat{p} + 1 \\
0 &= n\hat{p}^2 - 2\hat{p} + 1 - (n-1)u_{\mathcal{O}\mathcal{F}} \\
\hat{p} &= \frac{2 \pm \sqrt{4 - 4n(1-(n-1)u_{\mathcal{O}\mathcal{F}})}}{2n} \\
&= \frac{1 \pm \sqrt{1 - n(1 - nu_{\mathcal{O}\mathcal{F}} + u_{\mathcal{O}\mathcal{F}})}}{n} \\
\hat{p} &= \frac{1 + \sqrt{1 - n + n(n-1)u_{\mathcal{O}\mathcal{F}}}}{n}
\end{aligned} \tag{5.7}$$

We take only the positive square root because using the negative square root would give us a negative value for $\hat{p}$.

Figure 5.1 shows $\hat{p}$ as a function of $U(\mathcal{O}|\mathcal{F})$ for $n = 2$ and $n = 10$. $U(\mathcal{O}|\mathcal{F})$ has a maximum value of 1: this occurs when, for each value of $\mathcal{F}$, $\mathcal{O}$ always has the same value (i.e., $\mathcal{F} \succ \mathcal{O}$). In this case, $\hat{p} = 1$ as well (the most likely outcome for each $\mathcal{F}$ always occurs).

$U(\mathcal{O}|\mathcal{F})$ is minimized when $\mathcal{O}$ is uniformly distributed, regardless of $\mathcal{F}$; that is, when each possible outcome o_i is equally likely. In this case, if $\mathcal{O}$ takes on n different values, $U(\mathcal{O}|\mathcal{F}) = 1/n$ and $\hat{p} = 1/n$. One can see that this is the minimum value by requiring that the quantity under the square root in Equation 5.7 be positive; this minimum uniformity value explains why the curve in Figure 5.1 starts at $1/2$ for $n = 2$ and at $1/10$ for $n - 10$.

These two extreme cases represent the "most skewed" and "most even" distributions of outcomes. Any distribution that is not flat must be biased towards some values, and so will have a greater uniformity than a flat distribution. For these intermediate distributions, $\hat{p}$ will be greater than the uniformity.

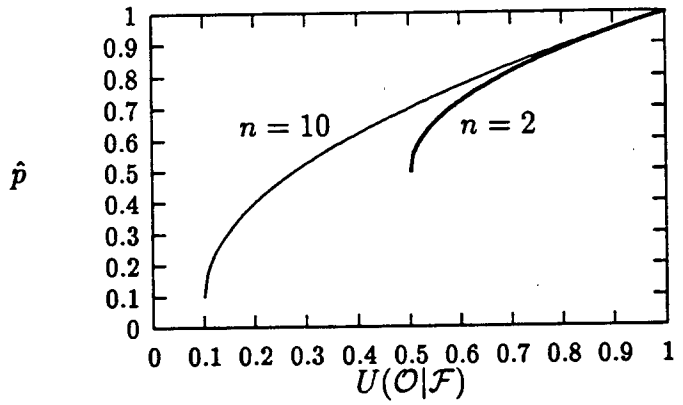

Figure 5.1: Expected Probability as a Function of Uniformity

5.5 Learning Curves

Results from computational learning theory suggest that the number of examples $\hat{m}$ needed to learn a theory is proportional to the Vapnik-Chervonenkis (V-C) dimension of the bias [Blumer *et al.*, 1986]. In other words,

$$\hat{m} = cd \tag{5.8}$$

where d is the V-C dimension of the bias, and c is a constant (which we approximate empirically).

A set of examples S is said to be *shattered* by a bias B if, given any division of S into positive and negative examples, there exists a theory in B that would so classify them. The V-C dimension of B is the size of the largest set of examples that can be shattered by B.

In the case of a decision tree (which is the representation used in ID*), every theory can be described, so the V-C dimension d is equal to the size of the space of possible examples. Therefore,

$$\hat{m} = cd = c \prod_{i=1}^{f} n_i \tag{5.9}$$

where f is the number of features in $\mathcal{F}$, and n_i is the number of values of feature i.

Bounds on learning curves (specifying error as a function of time) have been given by [Haussler *et al.*, 1990] for the deterministic case, and by [Haussler *et al.*, 1991] for the probabilistic case. Unfortunately, these bounds are only useful for large sample sizes, and we are interested in relatively small sample sizes. Also, it is not clear how to use the probabilistic bounds in practice.

In the current implementation, we make the simplifying assumption that prior to finding the best theory (for $t < \hat{m}$), predictions are no better than random guessing; after this, they are as good as we expect them to get (i.e., $\hat{p}$). Then the quality of predictions as a function of time will be

$$q(t) = \begin{cases} \frac{1}{n} & \text{if } t < \hat{m} \\ \hat{p} & \text{otherwise} \end{cases} \tag{5.10}$$

where $\hat{p}$ is as given in Equation 5.7. The learning curve is shown in Figure 5.2.

The actual learning curve will, of course, be smoother; however, in our preliminary tests (on relatively small, simple theory spaces), the bias value (i.e., average discounted accuracy) yielded by empirical tests is not too far from that predicted by this learning curve. For more accurate results, though, a better learning curve will be needed; unless better results from computational learning theory can be applied, the curve should be approximated empirically. In fact, it may be preferable to run empirical tests using the actual learning algorithm, in order to compute the expected accuracy of predictions using that particular learning algorithm.

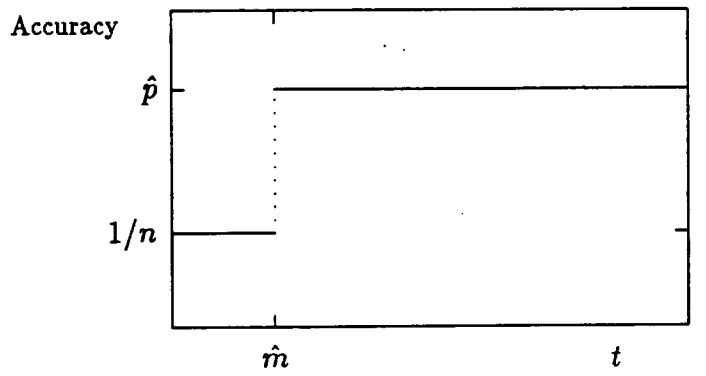

Figure 5.2: Quality of Predictions as a Function of Time

5.6 Time Preference Functions

The effect of the passage of time on the value of predictions depends on a variety of factors, including the amount of computation time available, what the prediction task is, the cost of making errors, the life expectancy of the agent, how fast the environment is changing, and how many other tasks need to be accomplished simultaneously with this learning task.

In PBE, the effects of these various factors are modeled as a time-preference function $\mathcal{T}(t)$. Time-preference functions are used in decision analysis [Holtzman, 1989] to indicate the degree to which the importance of reward on a prediction task changes over time (i.e., the degree of discounting of future rewards). If an agent's prediction task involves making a single prediction at time t_0, for example, only the accuracy of the agent's prediction at that moment matters: earlier and later performance is irrelevant. In this case, the time-preference function is zero at all points except for a spike at time t_0 (assuming there are no other relevant factors, such as other tasks that need to have time allocated to them).

A reasonable time-preference function for a simple autonomous agent, which is constantly making predictions in a dynamic environment, is γ^t, based on a constant discount rate γ, close to 1. PR uses this time-preference function with discount rate $\gamma = .8$.

Intuitively, using $\mathcal{T}(t) = \gamma^t$ means that accurate predictions in the distant future are exponentially less important than near-term accuracy; but any correct prediction, no matter how distant, has *some* positive value. The closer γ is to 1, the more heavily long-term performance is counted.

The value of γ will depend on the particular environment in which the agent finds itself, and should be determined experimentally (and ideally should be dynamically modifiable by the agent).

5.7 Expected Value of Biases

Combining the bias's accuracy over time (Equation 5.10) with the time-preference function $\mathcal{T}(t)$, and integrating over time, yields a general equation for the value of a bias:

$$V = \int_1^\infty q(t)\,\mathcal{T}(t)\,dt$$

Using the simplified learning curve from Equation 5.10 and letting $\hat{m} = cd$, where d is the V-C dimension of the bias, gives

$$V = \int_1^{cd} \frac{1}{n}\mathcal{T}(t)\,dt + \int_{cd}^\infty \hat{p}\,\mathcal{T}(t)\,dt$$

Using the time-preference function $\mathcal{T}(t) = \gamma^t$,

$$\begin{aligned} V &= \int_1^{cd} \frac{\gamma^t}{n}\,dt + \int_{cd}^\infty \gamma^t\,\hat{p}\,dt \\ &= \left[\frac{\gamma^t}{n\ln\gamma}\right]_1^{cd} + \left[\frac{\hat{p}\gamma^t}{\ln\gamma}\right]_{cd}^\infty \\ &= \frac{1}{\ln\gamma}\left[\frac{\gamma^{cd}-\gamma}{n} + \hat{p}(\gamma^\infty - \gamma^{cd})\right] \\ &= \frac{1}{\ln\gamma}\left[\gamma^{cd}(\frac{1}{n} - \hat{p}) - \frac{\gamma}{n}\right] \\ &= \frac{-1}{\ln\gamma}\left[\gamma^{cd}(\hat{p} - \frac{1}{n}) + \frac{\gamma}{n}\right] \end{aligned} \tag{5.11}$$

where $\hat{p}$ is as given in Equation 5.7. This is the bias evaluation function we use in PR. Notice that since γ and n are constant for a given learning task, the only term

that varies between biases is $\gamma^{cd}(\hat{p} - 1/n)$. Intuitively, if γ is large, so that the agent is willing to wait for accurate predictions, d has less influence on the value (in the extreme case, $\gamma = 1$ and $\gamma^{cd} = 1$ regardless of d). As $\hat{p}$ grows, the bias becomes more predictive, and the value of the bias increases.

5.8 Results

The effects of the cost associated with larger feature sets were measured using ID*, a probabilistic, incremental version of ID3 based on [Quinlan, 1986] and [Utgoff, 1988] (see Appendix B for a complete description of ID* and of the domain). ID* was run in a synthetic learning domain using various subsets of the full feature set as the learning bias; the results are given in the following section. We conclude that the cost of using a larger feature set can be prohibitively high, and that real-time agents acting in complex environments will have to address the tradeoff between time spent learning theories and their eventual accuracy.

5.8.1 Learning Procedure

The domain used for the tests described here consists of six descriptive features (shape, location, size, texture, smell and age, with (respectively) 4, 4, 3, 4, 5, and 6 values), and a goal descriptor (color) with 4 values.

The uniformities were fixed as follows:

$$\begin{aligned} U(\text{color}|\text{shape}) &= .58 \\ U(\text{color}|\text{location}) &= .81 \\ U(\text{color}|\text{size}) &= .27 \\ U(\text{color}|\text{texture}) &= .33 \\ U(\text{color}|\text{smell}) &= .25 \\ U(\text{color}|\text{age}) &= .27 \end{aligned}$$

The unconditional uniformity of color is .25 ($U(\text{color}) = .25$). That is, given no

information, each color is equally likely. Therefore, $U(\text{color}) = U(\text{color}|\text{smell})$ and "smell" has no predictive value.

Probabilities were generated to correspond to these uniformities (although more than one probability distribution is possible, the one used here was chosen to be fairly smooth). Examples were generated by selecting the goal (color) value randomly, then choosing values for the six descriptive features independently, according to the probability distribution. The process of generating the domain and examples is described in Appendix B.

In each test run, one hundred training examples were generated, and ID* was used to build four sets of decision trees:

- One using all six features.
- One using only texture (a relatively non-predictive feature).
- One using only location (a highly predictive feature).
- One using location and shape (the two most predictive features).

A set of forty test examples was generated from the same distribution; the four trees were tested on all forty examples after each training example. Additionally, the time spent processing each training example using each bias was measured.

Three test runs were performed and the results averaged. The average predictive quality n (number of test examples classified correctly) is shown as a function of the number of training examples in Figure 5.3. Figure 5.4 shows the average time spent processing each observation for each bias.

5.8.2 Analysis of Results

Given the uniformity values, the expected accuracy of each tested feature set can be computed using Equation 5.7:

$$\hat{p}_{\text{all}} \cong .98$$

$$\hat{p}_{\text{loc}} \cong .9$$

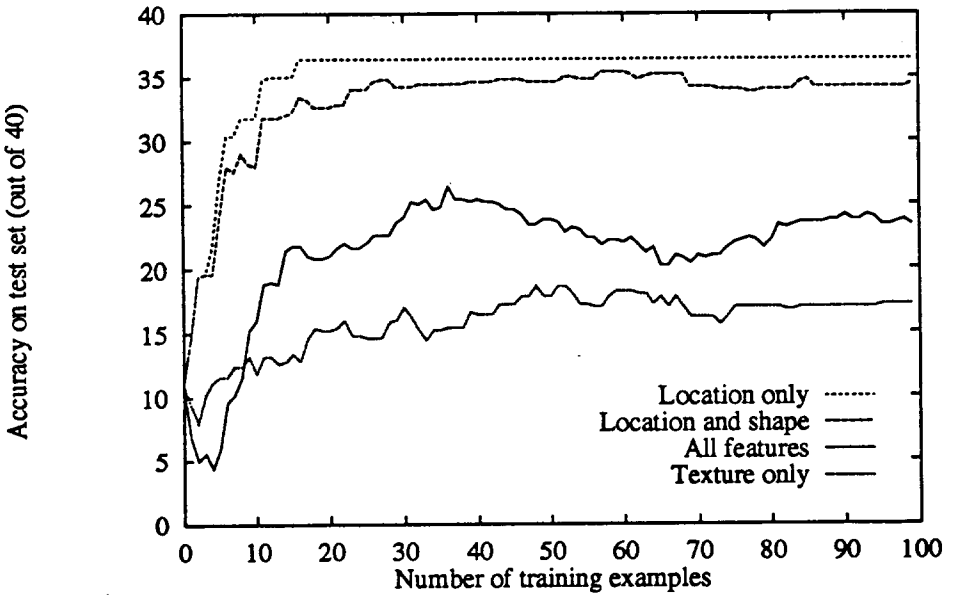

Figure 5.3: Results of Learning using 4 Different Biases

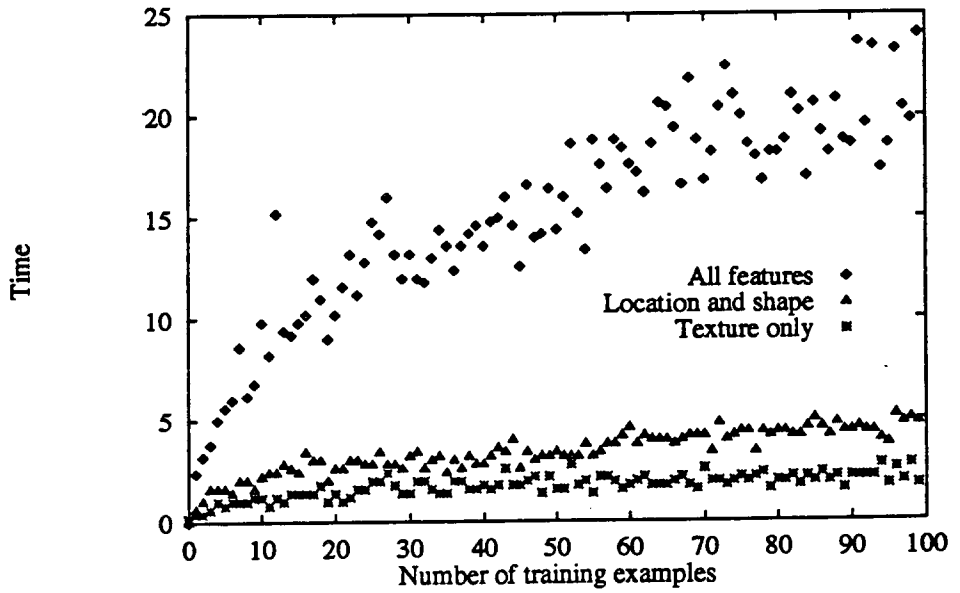

Figure 5.4: Time spent processing each observation

$$\hat{p}_{\text{loc-shape}} \cong .95$$

$$\hat{p}_{\text{texture}} \cong .5$$

Out of forty test examples, a tree built using these four biases on a large number of training instances would be expected to yield (on average) 39.2, 26, 38, and 20 correct predictions, respectively.

Location-only and texture are the only biases that performed approximately as well as expected, given the 100 training examples. Location-and-shape did almost as well, but the tree using all the features is lagging far behind. Clearly, the larger sample size needed for convergence in larger spaces is hindering its performance. Presumably, given enough training examples (and computation time), the bias using all the features would converge on nearly perfect prediction accuracy, but the marginal amount of accuracy gained over the predictions made by location only or location and shape is unlikely to be significant for many learning tasks. Considering that this is a relatively simple domain (compared to human learning domains), the degree to which the complete feature set is impaired is rather surprising.

The actual computation time spent processing each observation is shown in Figure 5.4. (The location-only data is not shown, since it closely matches that of texture-only.) The tree-building algorithm takes significantly longer to run using all the features than using the smaller feature sets. Also, the time seems to still be growing steadily after 100 examples.

If real-time behavior (in both learning and prediction) is needed, limiting the feature set will be a necessity. This tradeoff—of ultimate accuracy vs. time spent getting to that accuracy (and wrong predictions made in the meantime)—is captured by the time-preference curve described in Section 5.6.

The relative bias values for the domain are shown in Figure 5.5. (The $[-1/\ln\gamma]$ factor was left off for scaling purposes.) Location is the best choice unless γ is very high (location and shape outperform location only when γ is around .999). The improved accuracy of location and shape outweighs its expense compared to texture only when $\gamma \approx .95$. Using all of the features is not worthwhile unless γ is extremely close to 1.

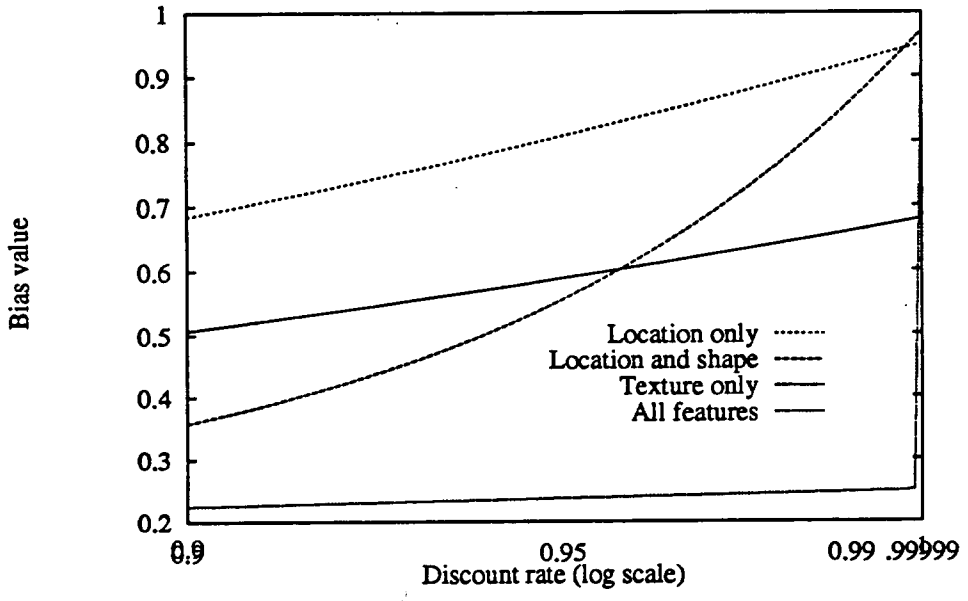

Figure 5.5: Relative bias values for the four feature sets tested

Intelligent resource-bounded agents clearly need to focus their learning mechanism in order to learn efficiently and quickly. However, controlling this focusing process requires examining the particular domain (learning task and problem environment). PBE provides a tool to do this by evaluating biases using domain-specific background knowledge (uniformities), agent-specific learning curves, and environment-specific time-preference functions.

Chapter 6

Probabilistic Learning

In a complex environment, no matter how good an agent is at forming theories, it will sometimes be unable to make predictions about the world with complete certainty. Uncertainty may arise from the environment, from the agent's sensory mechanisms, from the agent's internal processes, or from the agent's history of interaction with the world. Sources of uncertainty include:

Randomness: The world may be nondeterministic. In this case, even an omniscient observer would be unable to predict the world correctly.

Complexity: There may be a number of unlikely exceptions which would be expensive to enumerate, or the "true" theory of the world would take too long, or be too large, to learn precisely.

Representational limitations: The agent may be unable to express a correct deterministic theory (even if one exists) in its concept language.

Sensory limitations: The agent's senses only report a limited amount of information about the world. If important data are not reported, the agent may be unable to acquire the knowledge necessary to characterize the world precisely.

Sensory inconsistency: The agent's sensors may not always report the same perceived world in identical world states, due to noise in the sensors.

Insufficient sample size: If only a subset of the possible observations have been made, there may be multiple consistent hypotheses. In this case, the belief in in any one of them cannot be held with certainty.

These sources of uncertainty interact. For example, if the agent is unable to perceive relevant aspects of the environment (a sensory limitation) and the information it does receive from the environment is inconsistent (noise), the model it builds will be more distorted than the model built in the presence of only one of these handicaps.

If the world is chaotic (deterministic, but in such a way that outcomes are heavily dependent on initial conditions), the agent will be seriously handicapped by sensory limitations (inability to perceive initial conditions precisely), noise (incorrectly reported initial conditions), and complexity (if the agent's bounded resources are insufficient even in theory to compute the chaotic function to the necessary degree of precision).

In general, it will be impossible for the agent to be certain which of the sources of uncertainty are present. For example, to a finite agent, a complex enough world will appear nondeterministic simply because the agent is incapable of representing a deterministic world model, and will have to collapse some distinct states together in order to build a tractable (but nondeterministic) model.

In some cases, the agent may be able to tell after the fact that it has eliminated a source of uncertainty—for example, if it builds a tool (e.g., an infrared detector) to enable it to perceive an aspect of its environment that it was previously unable to detect directly, and the resulting model is better, a sensory limitation has been overcome. But it cannot know *a priori* which uncertainty sources are present.

In addition to being able to behave effectively in the presence of the above sources of uncertainty, there are functional advantages to using a probabilistic representation for theories. First, probabilistic theories are less brittle than deterministic theories: the behavior of the system degrades gracefully as the quality of the theory decreases. Second, *a priori* preferences (i.e., learning biases) can be expressed as prior probabilities, which are gradually overridden by data.

PAGODA uses statistical probabilities within the agent's theories to represent probabilities of outcomes given the external state of the world and the agent's actions; subjective probabilities are used to decide which world model (set of theories) is the most effective representation of the environment.

The statistical probabilities in PAGODA's theories represent summaries of observed frequencies of events. The probability of an outcome, given a conditioning context (consisting of a partial world state and an action taken by the agent in that state), is estimated by the ratio of the number of observations in which both the conditioning context and the outcome held to the total number of observations in which the conditioning context held.

Subjective probabilities are used to evaluate the agent's predictive theories. In order to decide which proposed theory is most effective, a Bayesian analysis is performed, combining a prior probability (based on the structure of the theory) with the likelihood of the evidence seen (which is computed using the statistical probabilities in the theories).

In this chapter, we describe PAGODA's method for learning probabilistic theories. Theories are represented as Uniquely Predictive Theories (UPTs; see Chapter 3) and are evaluated using a Bayesian method which is described in the following sections.

6.1 Theory Evaluation

In this section, we develop a Bayesian method for measuring the quality of proposed uniquely predictive theories (UPTs). As we will show, only two terms need to be considered: the accuracy of the theory, given by the likelihood of evidence $P(E|T)$, and the prior probability of the theory, $P(T)$. The former quantity is computed using PCI; we define the latter in terms of simplicity.

Section 6.1.1 discusses general issues involved in evaluating probabilistic theories. The formula used to evaluate theories is derived in Section 6.1.2, and the prior probability distribution used by PAGODA is given in Section 6.2. Finally, likelihood is computed in Section 6.3.

6.1.1 Evaluating Probabilistic Theories

In most models of concept learning, observations are assumed to be correct (i.e., noise-free) and consistent with some deterministic hypothesis which is expressible in the hypothesis language (for example, [Mitchell and Keller, 1983, Muggleton and Buntine, 1988, Kuipers, 1985, Carbonell and Gil, 1987]). The concept learning problem under this assumption becomes that of finding a hypothesis in the concept space which is consistent with all of the observed instances of the concept to be learned. Consistent hypotheses may be prioritized according to some preference metric (e.g., of simplicity or specificity), or they may all be considered equally good (as in the version space algorithm).

In a nondeterministic or noisy environment, we can no longer expect to find a completely consistent hypothesis. The problem then becomes that of finding the hypothesis which "best describes" the observed instances.

The question is, how do we define "best description?" If we allow enough parameters in the concept description, there will always be *some* theory that is consistent with all of the data (for example, we can just take the disjunction of all of the observations). The problem with this approach is that the resulting theory will be cumbersome and expensive to use, and is not likely to make any useful predictions (or perhaps not able to make any predictions at all). Of course, an agent can restrict the hypothesis space to theories that will be usable (see Chapter 5), but beyond that, it still needs a preference structure; otherwise, overfitted hypotheses (i.e., hypotheses fitted to noise) will be selected. The tradeoff can be thought of as one between simplicity and accuracy on the training set.

Using a simpler theory has two advantages. First, minimizing error on the training set may actually cause the theory to be fitted to noise, and therefore will not minimize future error. Second, even if the simpler theory is less accurate, the cost saved in applying it may outweigh the loss of accuracy for a limited rational agent.

The approach presented here is motivated on the one hand by algorithmic complexity and Minimum-Length Encoding (MLE) techniques [Solomonoff, 1964a, Solomonoff, 1964b, Chaitin, 1977] and on the other hand by Bayesian probability

theory and philosophical approaches to scientific theory formation using simplicity metrics (see, for example, [Good, 1983, Goodman, 1958]).

6.1.2 Bayesian Probability

Recall that the proposed theories (which must be UPTs) consist of conditional probabilities which are determined empirically. These probabilities *in* the theory are distinct from the probability *of* the theory. The goal in this section is to find the probability *of* a theory.

The theory with the highest probability should be that with the most effective structure for representing the observed data. Given this structure, the probabilities within the theory are straightforward to optimize. Complex structures (those with many dependencies) cost the agent in terms of space, computation time, and risk of overfitting. On the other hand, simple structures with only a few dependencies may not capture important relationships in the world.

The probability we wish to find, then, is the probability that the structure of this theory is the best representation of the behavior of the environment. It is *not* the probability that the particular values of the conditional (statistical) probabilities in the theory are correct, or even that they are close.[1] The statistical probabilities are estimated using observed frequences; this maximizes the accuracy of the theory as given by the Bayesian likelihood $P(E|T \wedge K)$.

Using the notation

T	a proposed theory
K	background knowledge
E	evidence: a sequence of observations $e_1, e_2, \ldots, e_n$

Bayes' rule gives

$$P(T|K \wedge E) = \frac{P(T|K)\,P(E|T \wedge K)}{P(E|K)} \tag{6.1}$$

[1] Which is not to say that the statistical probabilities *aren't* close, simply that we do not claim to measure their accuracy explicitly with this evaluation technique.

We are only interested in finding a relative probability in order to compare probabilities of competing theories, so the normalizing factor $P(E|K)$ in the denominator can be dropped,[2] yielding

$$P(T|K \wedge E) \propto P(T|K)\, P(E|T \wedge K) \tag{6.2}$$

We also assume that the individual observations $e_1 \ldots e_n$ composing E are independent, given K and T. This standard conditional independence assumption is reasonable, because the theories generated by the agent make independent predictions. Therefore, T embodies an assumption that the observations are independent, which must be true if T holds. Therefore,

$$P(T|K \wedge E) \;\propto\; P(T|K)\; \Pi_{i=1}^{n} P(e_i|T \wedge K) \tag{6.3}$$

The first quantity on the right-hand side represents the "informed prior"—i.e., the probability of the theory given the background knowledge K, but no direct evidence. The second quantity represents the likelihood of the theory, i.e., the combined probabilities of each piece of evidence given the theory and K.

6.2 Prior Probability

The prior probability of a theory, $P(T)$, is the probability of T before any evidence has been collected. A prior, however, is never completely uninformed: even before any direct observations about a particular learning task are made, an agent's past experience, available sensors, and internal representation will affect its disposition to believe a theory, and hence its prior probability distribution. For example, even if you have never been to a particular theater, your general background knowledge about theaters allows you to learn quickly how to buy tickets and refreshments and how to find your seat. All of the background knowledge available to an agent should ideally be reflected in its "prior."

[2]Note that if we drop these normalizing factors, we no longer have a true probability distribution. However, for notational convenience, I continue to call the resulting measure P.

PAGODA uses background knowledge in the form of uniformities (see Chapter 5) to select the language in which its theories are represented. The background knowledge K of Equation 6.3 consists of those uniformities and theories about other goals (which are irrelevant to the probability of this theory). The relevant knowledge in K is then just the set of uniformities, but these are implicit in the bias B, so that

$$P(T|K) = P(T|B)$$

We can think of $P(T|B)$ as being the prior, $P(T)$, implicitly conditioned by the bias.

A variety of justifications have been proposed for the use of simplicity as one test of the value of a theory. The justification one chooses depends on, and conversely influences, the model of simplicity used. These justifications include:

1. Occam's razor: always choose the simplest theory which is consistent with the data.
2. Simple theories are less expensive (in time and space) to learn and use.
3. The Minimum Length Encoding principle uses a formal argument to show that the shortest (i.e., simplest) theory explaining the data is the best. (See Chapter 2.)
4. Empirical evidence shows that people prefer simpler theories. (See [Medin *et al.*, 1987].)
5. Using a smaller hypothesis space implies that few hypotheses can be found that are as good as the proposed one. [Pearl, 1978]

There are a number of difficulties with the "traditional" view of simplicity (inasmuch as there *is* a traditional view, which, as can be seen from the above list of justifications, is not obvious). The most common reference to simplicity is Occam's razor, which tells us to select the most simple of the consistent theories. But this has two problems: first, it does not tell us what to do when we have a complex theory with high accuracy on the learning set and a simpler, but slightly less accurate, theory. (Or, rather, it *does* tell us what to do—we are to reject all inconsistent theories out

of hand, which doesn't seem reasonable.) Second, it does not provide a definition of simplicity.

In MLE, the length of the coded theory is used as a measure of its simplicity. Applying MLE directly would require finding the shortest code length with respect to a specified Universal Turing Machine (UTM). However, this is not computable, and only provides good answers in the limit (i.e., for large amounts of data). An alternative method, used by existing MLE applications, is to use a hand-generated encoding for the domain, or a general "optimal" code such as Huffman encoding.[3] However, a Huffman encoding does not take advantage of any structure in the domain: in most languages, terms are not generated independently (as a Huffman code assumes).

Good suggests using as a measure of simplicity a function of the probability that a theory appears in a language.

> Perhaps the best plan is to define the complexity of a theory... as minus the logarithm of the probability that the linguistic expression that describes the theory would occur in the language when the statistical properties of the language are specified up to say di-word frequencies. ([Good, 1983], p. 155.)

In other words, the encoding should take into account not just the probabilities of individual terms, but probabilities of pairs of terms. In some domains, however, even this will not be enough: structure in the language may have effects on larger groups of terms. Also, Good later says (p. 235)

> Perhaps the weights [of the terms] should be minus the logarithms of the frequencies of these *categories* of words (instead of using the frequencies of the individual words and symbols and such). This would reduce the problem to the specification of the categories.

That is, individual terms may not be the right level to consider for computing simplicity; rather, higher-level classifications should be used to evaluate a theory. These categories might be semantic groupings of similar words (e.g., color terms or people's names) or syntactic groupings (e.g., nouns or adjectives). In the RALPH world, categories might be objects in the world, in which case instances of `wall`, `nasty`, and so forth would be grouped together to determine frequencies.

[3] In a Huffman encoding, the length of each predicate is a function of its frequency.

This approach is similar to the method of evaluating theories by first examining the structure alone, and then filling in the best structure with the numbers that optimize overall probability. In this case, we determine the simplicity of a theory by examining its structure (which is defined by which categories of words it contains), and then measuring the contents of the structure (individual words) with the likelihood defined in Section 6.3.

We have used PAGODA as a testbed to experiment with several different metrics of simplicity, which we will discuss below. They differ in the level of classification (theories, rules, features, or terms) and in the method of finding probabilities of members of the appropriate class (the two methods used are the uniform distribution and a Huffman encoding scheme).

Our conclusion is that which metric of simplicity is "best" depends on the domain. Using simplicity as the prior in a Bayesian evaluation process means that any prior will converge to the correct theory, given enough data, as long as the prior does not assign zero probability to the correct theory. However, in order to perform well in any given domain, the language used and prior knowledge available should be used to choose the measure of simplicity (i.e., the prior).

In the remainder of this section we discuss four simplicity-based prior probability distributions used in PAGODA. The results of using these different priors in various RALPH domains are given in Chapter 7.

6.2.1 Uniform Distribution on Theories

Under this prior, equal probability is assigned to every theory. For infinite theory spaces, this results in an improper prior (all theories have zero prior probability), but since we are interested only in relative probabilities, we can ignore the prior probability term and simply choose the theory with maximum Bayesian likelihood $P(E|T)$. This procedure finds a theory that exactly fits the data, if one exists. In case of a tie (where two theories have equal likelihood) the shorter one will still be preferred (i.e., the theory with fewer rules or, if the theories being compared have the same number of rules, the theory with fewer terms).

Suppose PR has constructed the following two simple theories, T_1 and T_2.

$$\begin{aligned} T_1: \ & \Box \ \rightarrow_{.5} \ \Delta u(t+1,-10) \\ & \quad \rightarrow_{.5} \ \Delta u(t+1,90) \\ T_2: \ & \Box \rightarrow_{1.0} \Delta u(t+1,-10) \\ & \texttt{action}(t,\texttt{:munch}) \rightarrow_{1.0} \Delta u(t+1,90) \end{aligned}$$

Suppose further that the evidence used to construct these theories consists of two observations:

$$\begin{aligned} e_1 &= \texttt{action}(1,\texttt{:munch}) \wedge \Delta u(2,90) \\ e_2 &= \texttt{action}(2,\texttt{:move-forward}) \wedge \Delta u(3,-10) \end{aligned}$$

The uniform distribution on theories assigns the same probability to the two theories ($P(T_1) = P(T_2)$). The likelihood of the two theories is simply the conditional probability of the evidence, given the theories:

$$\begin{aligned} P(E|T_1) &= P(e_1|T_1)P(e_2|T_1) = \tfrac{1}{2} * \tfrac{1}{2} = \tfrac{1}{4} \\ P(E|T_2) &= P(e_1|T_2)P(e_2|T_2) = 1 * 1 = 1 \end{aligned}$$

The Bayesian evaluation formula (Equation 6.3) gives

$$\begin{aligned} P(T_1|E) &\propto P(T_1)P(E|T_1) \propto \tfrac{1}{4} \\ P(T_2|E) &\propto P(T_2)P(E|T_2) \propto 1 \end{aligned}$$

Therefore, under the uniform distribution on theories, T_2 is preferable given the evidence.

6.2.2 Rule-level Classification of Theories

The level of classification under this distribution is rules in the theory. If N_R is the number of rules in the theory T, then the prior probability of T according to this distribution is

$$P(T) = \frac{1}{2^{N_R}}$$

This gives the prior probability that the correct theory has N_R rules; i.e., the probability of the *class* of T being the correct class. In this prior, the theory class can

be thought of as being generated by coin-flipping. If the coin comes up heads, we generate one final rule and stop; if it comes up tails, we generate another rule and continue flipping. Within a class (i.e., theories with the same number of rules), the particular rules are chosen to optimize the likelihood of the theory.

In the example given in the previous section, the rule-level distribution assigns prior probabilities

$$\begin{aligned} P(T_1) &= \tfrac{1}{2} \\ P(T_2) &= \tfrac{1}{4} \end{aligned}$$

since T_1 has one rule, and T_2 has two rules. The likelihoods are the same, so the relative posterior probabilities of the theories are

$$\begin{aligned} P(T_1|E) &\propto \tfrac{1}{2} * \tfrac{1}{4} = \tfrac{1}{8} \\ P(T_2|E) &\propto \tfrac{1}{4} * 1 = \tfrac{1}{4} \end{aligned}$$

T_2 is still preferable, but not by as much as under the uniform distribution on theories.

6.2.3 Feature-level Classification of Theories

This prior is similar to the rule-level classification of theories, but at a slightly lower level. The classification level is descriptive features in the conditioning context of rules; the probability distribution assumes that a coin is flipped to generate features. The probability of individual features is ignored. If N_F is the number of feature descriptors in the theory,

$$P(T) = \frac{1}{2^{N_F+1}}$$

Again, this represents the probability that the theory's class is the correct one; i.e., the probability that the correct theory has N_F features.

T_1 has no terms (since the only conditioning context is empty); T_2 has one term (`action`(t, `:munch`)). In the example, the feature-level distribution assigns prior probabilities

$$\begin{aligned} P(T_1) &= \tfrac{1}{2} \\ P(T_2) &= \tfrac{1}{4} \end{aligned}$$

In this case, the priors are the same as for the rule-level classification, so the resulting relative probabilities are the same, and T_2 is again preferred slightly.

6.2.4 Huffman Encoding of Terms

The classification level for this prior is terms ("words" within feature descriptors), but a uniform distribution is not assumed. Rather, we use the frequency of terms within the theory to compute an optimal encoding for the theory, and use the length of the encoded theory as the negative logarithm of its probability. If N_T is the number of terms in the theory, t_i is the ith term, and $\text{prob}(t_i)$ is the relative frequency with which t_i appears in the theory,

$$P(T) = \prod_{i=1}^{N_T} \text{prob}(t_i)$$

This approach is similar to that of Muggleton [1988], who uses the Minimum Length Encoding approach to decide which theories should be presented to an oracle (but does not use the code length to define a prior probability). If S consists a theory plus a set of data explained by the theory, Muggleton's technique approximates the length of an efficient encoding M of S as

$$|M(S)| \approx -N \sum_{s \in sym(S)} p_s \log_2 p_s \text{ bits}$$

where $sym(S)$ is the set of symbols in S, N is the number of different symbols found in S, and p_s is the relative frequency of s in S (i.e., $n_s/|S|$).

The frequencies of terms in the theories in the example given above are:

T_1 :	Δu	2
	$<$ *variable* $>$	2
	$<$ *number* $>$	2
T_2 :	Δu	2
	$<$ *variable* $>$	3
	$<$ *number* $>$	2
	`action`	1
	`:munch`	1

Variables and numbers are still classified together (i.e., treated as equivalent for the purposes of determining frequencies).

The prior probabilities of the theories in the Huffman encoding are therefore

$$\begin{aligned} P(T_1) &= \tfrac{1}{6}^6 &&= 2.14 \times 10^{-5} \\ P(T_2) &= \tfrac{2}{9}^2\tfrac{3}{9}^2\tfrac{2}{9}^2\tfrac{1}{9}\tfrac{1}{9} &&= 3.7 \times 10^{-7} \end{aligned}$$

T_1, the simpler theory, is preferred very strongly (by almost two orders of magnitude) by the Huffman encoding prior.

6.2.5 Expected Effects

The effects of the various priors in empirical tests are discussed in Chapter 7; below is a brief summary of the expected effects.

1. The uniform distribution on theories should generate very specific theories (i.e., relatively large theories with many rules). The performance of these theories should only be reasonable in fairly deterministic worlds where everything that happens is relevant to the outcome.

2. The rule-level distribution ($P(T) = 1/2^{N_R}$) should perform approximately as well as the uniform distribution on theories in simple worlds. However, in more complex, nondeterministic domains we expect that it will tend to overfit (i.e. to form theories that "explain" genuine randomness in the domain).

3. The feature-level distribution ($P(T) = 1/2^{N_F}$) should work reasonably well in simple nondeterministic environments, where there is a significant degree of randomness in results, but most properties that the agent observes affect the degree of randomness.

4. Huffman encoding of terms provides a very strong bias for simplicity. However, it will prefer theories in which all of the rules refer to the same one or two features, since it encodes the repeated features more compactly. Therefore, we expect that it will perform best in complex domains where only a few features are relevant.

Given enough data, and a good search mechanism, using the Bayesian method with *any* of these priors will converge to a maximum likelihood theory. However, in the

short term, knowledge about the complexity of the domain should be used to select a good starting point.

Additionally, in learning systems that use an incremental search heuristic (as does PAGODA), generating many bad (overly specific) theories early on can mean that the agent may never, in fact, find a good theory, because it can get stuck on a local maximum. From this we conclude that a good search heuristic should include techniques for "jumping" away from local maxima.

6.3 Likelihood of Evidence

PAGODA generates its own learning instances from the sequence of sensory inputs and actions, using the bias for each learning task. Each set of features that matches (unifies against) the conjunction of a learning goal and the bias is treated as an instance. For example, if the learning bias for $\Delta\mathtt{u}(t+1, u)$ is

$$\mathtt{action}(t, a)$$

and the agent's sequence of sensory inputs and actions includes (among other perceptions)

$$\mathtt{action}(0, \mathtt{:move\text{-}forward}) \wedge \Delta\mathtt{u}(1, -10) \wedge \mathtt{action}(1, \mathtt{:turn\text{-}right})$$
$$\wedge\ \mathtt{action}(2, \mathtt{:munch}) \wedge \Delta\mathtt{u}(2, -10) \wedge \mathtt{action}(3, \mathtt{:turn\text{-}left}) \wedge \Delta\mathtt{u}(3, 90)$$

then it will construct three instances for the learning goal $\Delta\mathtt{u}$:

$$\mathtt{action}(0, \mathtt{:move\text{-}forward}) \wedge \Delta\mathtt{u}(1, -10)$$
$$\mathtt{action}(1, \mathtt{:turn\text{-}right}) \wedge \Delta\mathtt{u}(2, -10)$$
$$\mathtt{action}(2, \mathtt{:munch}) \wedge \Delta\mathtt{u}(3, 90)$$

$P(e_t|T \wedge K)$ is the probability of the direct observation made at time t, given the theory and background knowledge. $P(e_t|T \wedge K)$ is equal to $P(e_t|T)$ if e_t is conditionally independent of K given T, which is a reasonable assumption since the theories in K make no predictions regarding T's goal G, and any relevant information in the uniformities in K has already been used to select the current bias.

If the theory being evaluated predicts Δu, e_t can be rewritten as

$$\texttt{senses}_t \wedge \texttt{action}_t \wedge \Delta u_{t+1}$$

This is because in the current implementation of PAGODA, only features at time t are considered for predicting features at time $t+1$. This assumption does not affect the analysis, though; for example, features at time $t-1$ could be included without any significant modifications. The probability of e_t, given the theory T, is

$$\begin{aligned} & P(\texttt{senses}_t \wedge \texttt{action}_t \wedge \Delta u_{t+1} | T) \\ & = P(\texttt{senses}_t \wedge \texttt{action}_t | T)\, P(\Delta u_{t+1} | \texttt{senses}_t \wedge \texttt{action}_t \wedge T) \end{aligned}$$

Since T makes no predictions regarding $\texttt{senses}_t$ and $\texttt{action}_t$, the first term can be rewritten as the prior probability:

$$P(\texttt{senses}_t \wedge \texttt{action}_t | T) = P(\texttt{senses}_t \wedge \texttt{action}_t)$$

We drop this term, since only the relative probability of theories is of interest and the value of this term will be the same for all theories on the same learning goal. The second term is computed by applying PCI (Chapter 3) to the theory, as we demonstrate below.

Using a bias for Δu which includes all sensory inputs and the action in a world containing nasties and food, PR might form the theory in Figure 6.1. Suppose that at time 5 PR's perceived world is:

$$\texttt{nasty-smell}(5,20) \wedge \texttt{food-smell}(5,10) \wedge \texttt{vision}(5,\texttt{wall},2) \wedge \Delta u(5,-10)$$

and it chooses the action :munch, resulting in $\Delta u(6,-10)$. The observation constructed is

$$\begin{aligned} e_5 = \texttt{nasty-smell}(5,20) & \wedge \texttt{food-smell}(5,10) \wedge \texttt{vision}(5,\texttt{wall},2) \\ & \wedge \Delta u(5,-10) \wedge \texttt{action}(5,\texttt{:munch}) \wedge \Delta u(6,-10) \end{aligned}$$

To compute the possible outcomes at time 6, PCI is applied. The most-specific rules for the sensory inputs at time 5, which in this case are the second and

$\Box$	$\rightarrow_{.6}$	$\Delta u(t+1,-10)$
	$\rightarrow_{.1}$	$\Delta u(t+1,-11)$
	$\rightarrow_{.2}$	$\Delta u(t+1,-60)$
	$\rightarrow_{.1}$	$\Delta u(t+1,90)$
action$(t,$:munch)	$\rightarrow_{.6}$	$\Delta u(t+1,90)$
	$\rightarrow_{.1}$	$\Delta u(t+1,-60)$
	$\rightarrow_{.3}$	$\Delta u(t+1,-10)$
nasty-smell$(t,20)$	$\rightarrow_{.4}$	$\Delta u(t+1,-10)$
	$\rightarrow_{.5}$	$\Delta u(t+1,-60)$
	$\rightarrow_{.1}$	$\Delta u(t+1,-11)$
action$(t,$:move-forward$) \wedge$ nasty-smell$(t,20)$	$\rightarrow_{.7}$	$\Delta u(t+1,-60)$
	$\rightarrow_{.3}$	$\Delta u(t+1,-10)$

Figure 6.1: Theory for a RALPH world

third rules in Figure 6.1, are combined using PCI, yielding:

$$P(\Delta u(6,90)|e_5)$$
$$= \frac{P(\Delta u(6,90)|\texttt{action}(5,:\texttt{munch}))\,P(\Delta u(6,90)|\texttt{nasty-smell}(5,20))}{P(\Delta u(6,90))} = \frac{.6*0}{.1} = 0$$
$$P(\Delta u(6,-11)|e_5)$$
$$= \frac{P(\Delta u(6,-11)|\texttt{action}(5,:\texttt{munch}))\,P(\Delta u(6,-11)|\texttt{nasty-smell}(5,20))}{P(\Delta u(6,-11))} = \frac{0*.1}{.1} = 0$$
$$P(\Delta u(6,-10)|e_5)$$
$$= \frac{P(\Delta u(6,-10)|\texttt{action}(5,:\texttt{munch}))\,P(\Delta u(6,-10)|\texttt{nasty-smell}(5,20))}{P(\Delta u(6,-10))} = \frac{.3*.4}{.6} = .2$$
$$P(\Delta u(6,-60)|e_5)$$
$$= \frac{P(\Delta u(6,-60)|\texttt{action}(5,:\texttt{munch}))\,P(\Delta u(6,-60)|\texttt{nasty-smell}(5,20))}{P(\Delta u(6,-60))} = \frac{.1*.5}{.2} = .25$$

The probabilities are normalized by PCI so that the probabilities of all outcomes sum to 1, yielding

$$\begin{aligned} P(\Delta u(6,90)|e_5) &= 0 \\ P(\Delta u(6,-11)|e_5) &= 0 \\ P(\Delta u(6,-10)|e_5) &= 4/9 \\ P(\Delta u(6,-60)|e_5) &= 5/9 \end{aligned}$$

The likelihood of e_5 is

$$P(e_5|T) = P(\Delta u(6,-10)|T \wedge \texttt{nasty-smell}(5,20) \wedge \texttt{food-smell}(5,10) \wedge \texttt{vision}(5,\texttt{wall},2) \wedge \Delta u(5,-10)) = 4/9$$

In order to operate in complex domains, intelligent agents must be able to cope with uncertainty arising from the environment and from the agent's physical and computational limitations. PAGODA's learning model addresses this necessity by using a probabilistic representation and inference method for theories, and Bayesian evaluation methods for learning the theories. Various simplicity metrics have been proposed; the degree to which the agent should bias its learning towards simpler theories depends on the complexity of the domain.

Chapter 7

Implementation and Results

PAGODA has been implemented as an agent in the RALPH (Rational Agent with Limited Processing Hardware) world. In this chapter we discuss the PAGODA implementation and present and analyze the results of running PAGODA in a number of RALPH worlds under varying conditions.

7.1 PAGODA-RALPH Implementation

This section describes the implementation of PR: first the high-level behavior of the system is described, then the processes of bias evaluation, hypothesis generation and testing, planning, and goal generation are presented.

7.1.1 Overall Behavior

The design of PAGODA was discussed in Section 1.3. The implementation of this design consists of three primary routines: `init-learning()` and PR's `:perform` and `:choose-action` methods.

`init-learning()` initializes the global variables containing PR's history, and initializes the set of learned theories to be empty. It also selects a bias for each initial learning goal using PBE (normally, the only initial learning goal is Δu).

PR's `:perform` method is called during each time slice by the scheduler. It

calls the learning routines described in Section 7.1.3 to update the current theory for each learning goal.

The `:choose-action` method calls the probabilistic planning mechanism described in Section 3.5 to select an action to be executed during the next time slice.

7.1.2 Bias Evaluation

PR is provided with background knowledge in the form of uniformities for each RALPH world. These uniformities were estimated based on the actual behavior of the world. They take into account the random behavior of the nasties, the complexity of the world, and PR's limited sensory inputs. All of them are of the form

$$U(S_2(t+1)|S_1(t)) = u \tag{7.1}$$

where S_1 and S_2 are subsets of PR's sensory inputs at the specified times. The only biases considered are those provided directly by the uniformities; in other words, only sets of features that appear in the conditioning context of some uniformity are evaluated. Therefore, no features from time $t-1$ are considered when predicting the environment at time $t+1$. This provides the agent with a strong time contiguity bias (i.e., a prior belief that events in the world are caused by immediately previous events). This bias could be relaxed by the designer by providing uniformities that include earlier events; the agent could automatically relax the bias by chaining uniformities together or by generating new terms that represent intermediate states, and using these to make predictions. The latter two possibilities will be discussed in Chapter 9.

A bias for each learning goal is selected by finding those uniformities which are relevant to the learning goal (i.e., where the goal is part of S_2 in Equation 7.1), computing the value of the bias represented by each uniformity (the set of features in S_1) using PBE (Chapter 5), and selecting the bias with the highest value.

The bias-value computation assumes that all features are Boolean (i.e., take on only two values), since it is not currently provided with any knowledge about the features. Since most features actually take on more than two values, this tends to assign slightly higher bias values to larger biases than they would otherwise receive.

The value of γ (the discounting rate) is .8. c, the learning constant in Equation 5.8, is 1.

7.1.3 Hypothesis Generation and Evaluation

The Bayesian method used by PAGODA for evaluating theories was presented in Chapter 6. This section describes the heuristic search process used to find new theories to evaluate. The search is incremental, in the sense that as each new observation arrives, new theories are generated. However, all of the previous observations are stored in the theories, so the search process is not bounded in space or time.

The search process can be described as a generate-and-test search through the space of possible theories, with an open list of fixed size n. After each new observation arrives, the observation is first incorporated into each of the theories on the open list. This is done by adding the observation to each of the most-specific rules (MSRs) in the theory that apply to it, and updating the probabilities in the theory accordingly. The search then uses the updated theories as seeds, and generates all neighboring theories, using a set of search operators which are described below. Theories that are not potentially better than the seed that generated them are pruned; the others are recursively used as seeds to generate more new theories. After the search ends—i.e., when no more potentially better neighbors can be found—the remaining theories are all evaluated, and the n best theories are retained for the next cycle. (In the current implementation, $n = 3$.)

A theory T_1 is considered to be potentially better than another T_2 if the probability of the observation given T_1 is greater than or equal to the probability of the observation given T_2. Because two theories can each be potentially better than the other by this definition, the search process can loop. Therefore, if a theory with the same structure as an existing theory is generated, the search along that path terminates, so that an infinite loop will not occur.

The first time the learning module is called for a particular learning goal, a single most-general theory is generated, stating that the specified outcome for that goal always occurs. This is always the theory with the highest probability, since it is

the simplest possible theory with perfect accuracy on the training set.

`Heuristic-Search:`

1. Insert the new observation into each seed theory.
2. Apply `Apply-operators` to each seed theory.
3. Apply `Generate-new-rules` to each seed theory.
4. Apply `Apply-operators` to each theory yielded by step (3).
5. Apply the Bayesian evaluation mechanism to all of the theories that were generated.
6. Return the n best theories.

`Apply-operators:`

For each MSR that made an incorrect prediction:

1. For each parent p of r, call `Try-merging` on p and r.
2. For each sibling s of r, call `Try-merging` on s and r.
3. For each child c of r, call `Try-merging` on c and r.

`Try-merging (r1,r2):`

1. Generate a theory that replaces `r1` and `r2` with a minimally more general rule `r`.
2. If the probability of the observation given `r` is greater than the probability of the observation given `r1` and `r2`, call `Apply-operators` on the new theory.

Figure 7.1: Heuristic search algorithm

There are four search operators:

`Generate-new-rules`: A set of minimally more specific rules is generated for each MSR in the seed theory that makes an incorrect prediction,[1] by specializing each feature that has a more general value in the MSR than in the observation. These features represent distinctions which may differentiate the outcomes. The instances which were covered by the MSR are divided between the MSR and the new rule. A new theory is generated with each of the new rules.

`Merge-into-parent`: Each MSR that made an incorrect prediction is merged into each of its parents, generating a set of theories with fewer rules than the seed theory.

[1] Recall that a probabilistic prediction is incorrect if the observed outcome is not the most likely outcome.

`Merge-with-sibling`: Each MSR that made an incorrect prediction is merged with each of its siblings, generating a set of theories with one rule that is more general than either sibling.

`Merge-with-child`: Each MSR that made an incorrect prediction is merged with each of its children, generating a set of theories with fewer rules than the seed.

All of the theories generated by these operators are used as seeds for further searching only if the resulting (new or merged) rule assigns at least as high probability to the current observation as the rule it was generated from.

The search algorithm is given in Figure 7.1.

7.1.4 Goal Generation

The current form of GDL (Chapter 4) finds the utility of learning goals, but does not account for the costs of learning. Therefore, the agent can only decide which learning goals are best, but not whether it is worth adding any learning goals at all. Because of this, the goal generation process is guided by the user. At any point, the user can request that the system evaluate all potential learning goals, and may then tell the agent to add the best of these goals to its set of learning goals; the new goals will be used in future learning and planning.

7.2 Tests and Results

Two RALPH worlds were used for these tests: a small world containing only food, and a larger world containing nasties. In the small-food world (Figure 7.2), whenever PR manages to consume all of the food, a new cupcake appears somewhere else in the world. In the nasties world (Figure 7.3), food is not regenerated (none of the tests were long enough for PR to consume all of the food) but whenever PR zaps a nasty, a new nasty appears elsewhere, so that two nasties are always present. Each

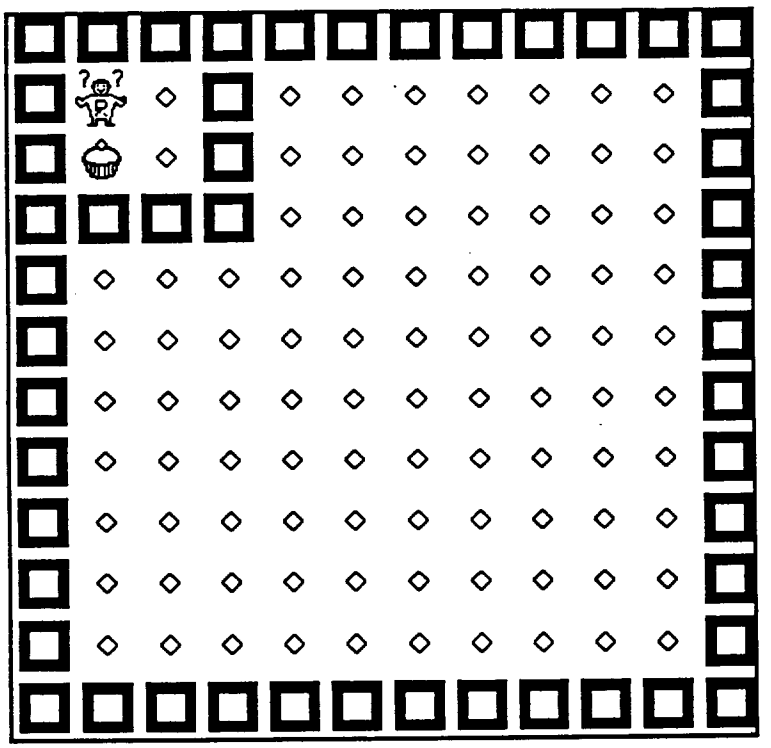

Figure 7.2: Small-food test world

time the nasties world is initialized, the objects present in the world (food, walls and nasties) are distributed randomly.

Three measures are used to evaluate the system's performance:

- Accuracy on a test set: before each test is started, the world is run, guiding the agent by hand to ensure that a representative set of observations will be generated; these observations are used as a test set. Accuracy on the test set measures generalization ability, and is used as a primary measurement of PR's behavior. The same test set is used for all measurements in each test.

- Time spent to process each observation: this is the internal run time of the system while the learning module is active. This measures the complexity of the learning task for a particular bias and domain.

- Success on the agent's task: the average utility per unit time is used as a measure of the agent's success. Since the agent's goal is to maximize its utility, and the

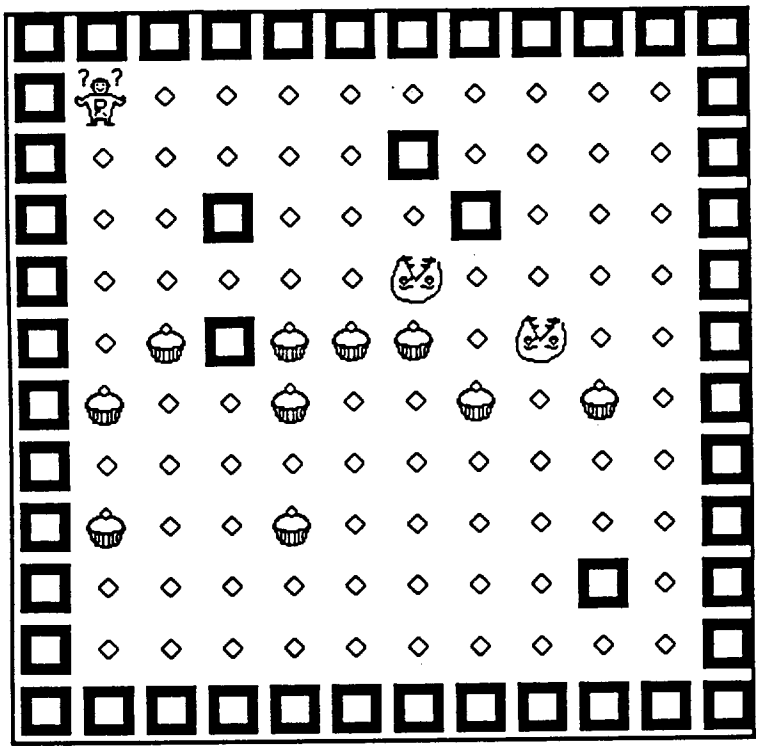

Figure 7.3: Nasties test world

purpose of learning is to further that goal, the utility earned by the agent tests the system as a whole. Utility is used to evaluate the effects of adding new learning goals to the system.

Section 7.2.1 presents a set of tests showing the effect of varying the bias selection procedure, and demonstrating the overall learning capability. The tests in Section 7.2.2 show the performance of PAGODA when different prior probability distributions are used for learning. Section 7.2.3 demonstrates the goal evaluation mechanism and shows the improvement in the system's behavior resulting from adding learning goals to the system. Overall conclusions are given in Section 7.3.

7.2.1 Testing Biases and Learning

The tests in this section demonstrate the effects of bias selection and show the overall learning performance of the system. In the first two tests, the feature

set selected by PBE was compared to two alternatives: using all available features (Δu, `nasty-smell`, `food-smell`, and `vision`), and using no features (i.e., generating a single theory with one default rule). These three biases were run in the two test worlds. In the third test, a single scenario (using all of the features in the nasties world) was run twice, to show the variability of performance resulting from the agent's experiences in different runs. This test highlights the difficulty of learning at all in this domain, and of getting consistent test results.

For the tests in this section and in Section 7.2.3, the rule-level prior probability distribution was used. For all of the tests in this chapter, PR was guided by hand for the first few steps to a node containing food, and then allowed to run autonomously. Learning occurs whether PR chooses its own actions or is guided by hand.

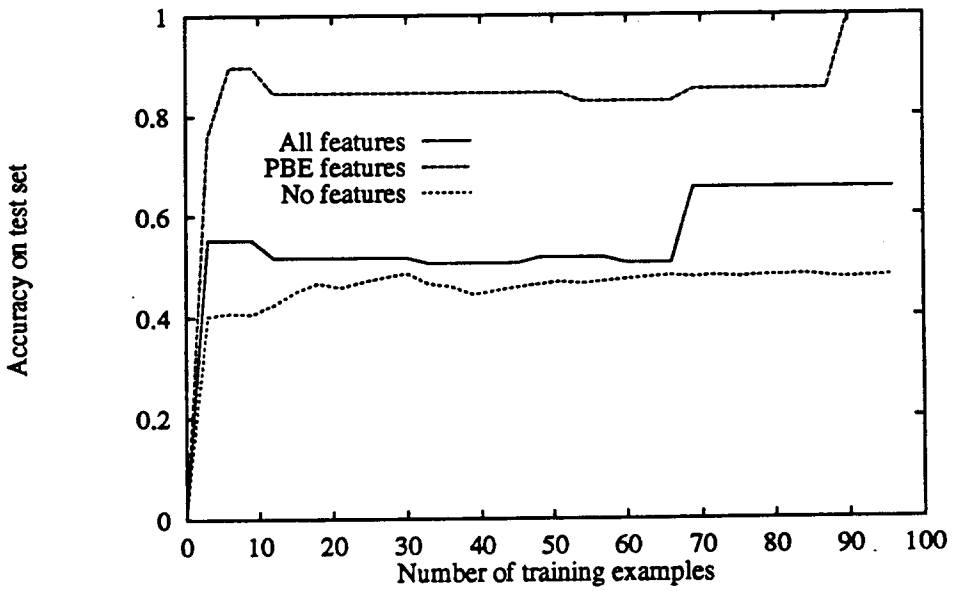

Figure 7.4: Bias tests in small food world: accuracy measurements

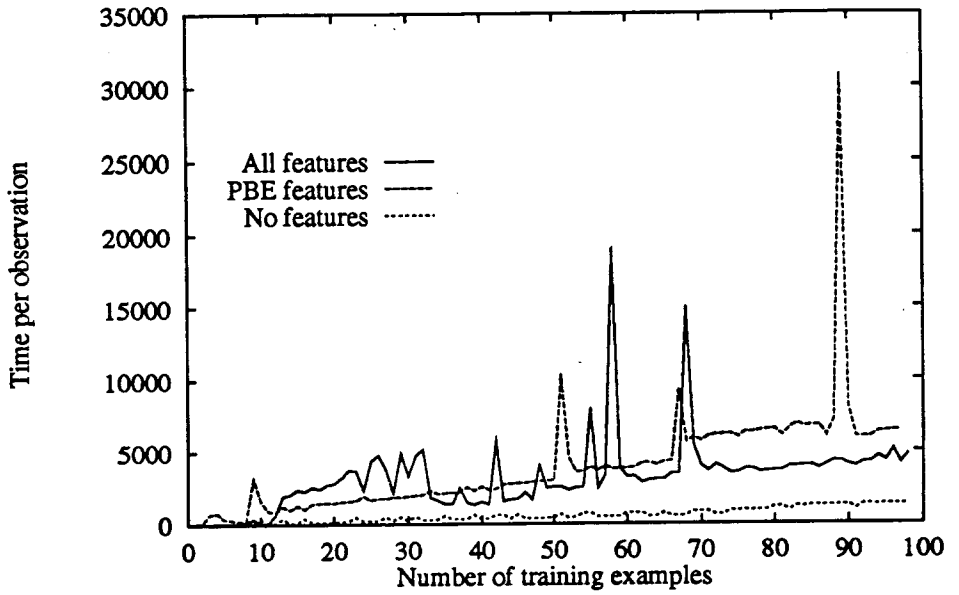

Figure 7.5: Bias tests in small food world: time measurements

Small-food world

For the first test, PR was run in the small-food world. The uniformities for Δu in this world are as follows:

$$U(\Delta \mathtt{u} | \mathtt{vision} \wedge \mathtt{food\text{-}smell} \wedge \mathtt{action}) = 1.0$$
$$U(\Delta \mathtt{u} | \mathtt{vision} \wedge \mathtt{food\text{-}smell} \wedge \mathtt{nasty\text{-}smell} \wedge \mathtt{action}) = 1.0$$

The bias values assigned by PBE are:

$$V(\mathtt{vision} \wedge \mathtt{food\text{-}smell} \wedge \mathtt{action}) = 4.73$$
$$V(\mathtt{vision} \wedge \mathtt{food\text{-}smell} \wedge \mathtt{nasty\text{-}smell} \wedge \mathtt{action}) = 4.50$$

therefore vision $\wedge$ food-smell $\wedge$ action is selected as the learning bias.

The results of this test are shown in Figures 7.5 and 7.4. Using the PBE features gave good results, averaging an accuracy of .85 over the run, and eventually

converging to 1.0. The theory learned is

$$\Box \rightarrow_{1.0} \Delta\mathrm{u}(t+1, -10)$$
$$\texttt{vision}(t, \texttt{wall}, 1) \wedge \texttt{action}(t, \texttt{:move-forward}) \rightarrow_{1.0} \Delta\mathrm{u}(t+1, -11)$$
$$\texttt{food-smell}(t, 20) \wedge \texttt{action}(t, \texttt{:munch}) \rightarrow_{1.0} \Delta\mathrm{u}(t+1, 90)$$

which is, in fact, a correct theory for the domain: eating when `food-smell` is 20 gives 90 utility "points;" moving forward into a wall causes the agent to lose 11 points; all other actions result in a loss of 10 points.

Using all of the features resulted in lower performance, with a maximum accuracy of .65. The no-feature bias performed poorly, yielding only 0.5 accuracy.

As expected, the no-feature bias takes less time than either PBE or all-features. PBE actually seems to take longer on average than all-features, but both have a number of spikes. The largest spike appears in the PBE bias run at the same time as the observation that PR used to generate the correct theory (t=90). Another, smaller spike appears at the same time that all-features jumps to .65 accuracy (t=69). This weak correlation of time spikes and theory shifts appears in some of the later tests as well, but the underlying explanation of why the heuristic search generates these spikes is unknown.

Nasties world

The uniformities for Δu in the nasties world are:

$$U(\Delta\mathrm{u}|\texttt{vision} \wedge \texttt{food-smell} \wedge \texttt{nasty-smell} \wedge \texttt{action}) = .8$$
$$U(\Delta\mathrm{u}|\texttt{vision} \wedge \texttt{food-smell} \wedge \texttt{nasty-smell} \wedge \Delta\mathrm{u} \wedge \texttt{action}) = .9$$

The resulting bias values are:

$$V(\texttt{vision} \wedge \texttt{food-smell} \wedge \texttt{nasty-smell} \wedge \texttt{action}) = 4.30$$
$$V(\texttt{vision} \wedge \texttt{food-smell} \wedge \texttt{nasty-smell} \wedge \Delta\mathrm{u} \wedge \texttt{action}) = 4.24$$

The bias selected is `vision` $\wedge$ `food-smell` $\wedge$ `nasty-smell` $\wedge$ `action`. The accuracy on the PBE run is slightly higher than all-features; however, they do end up with approximately equal accuracy (Figure 7.6).

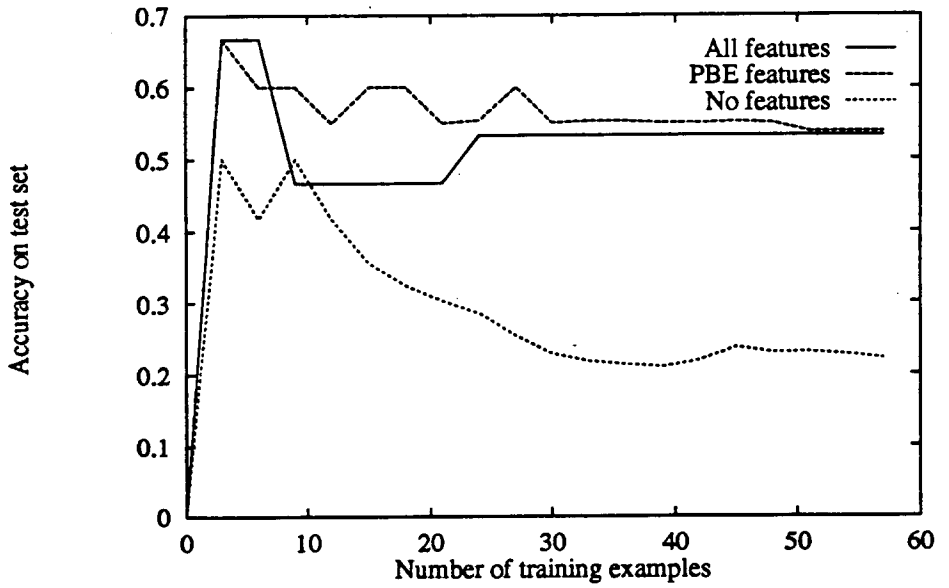

Figure 7.6: Bias tests in nasties world: accuracy measurements

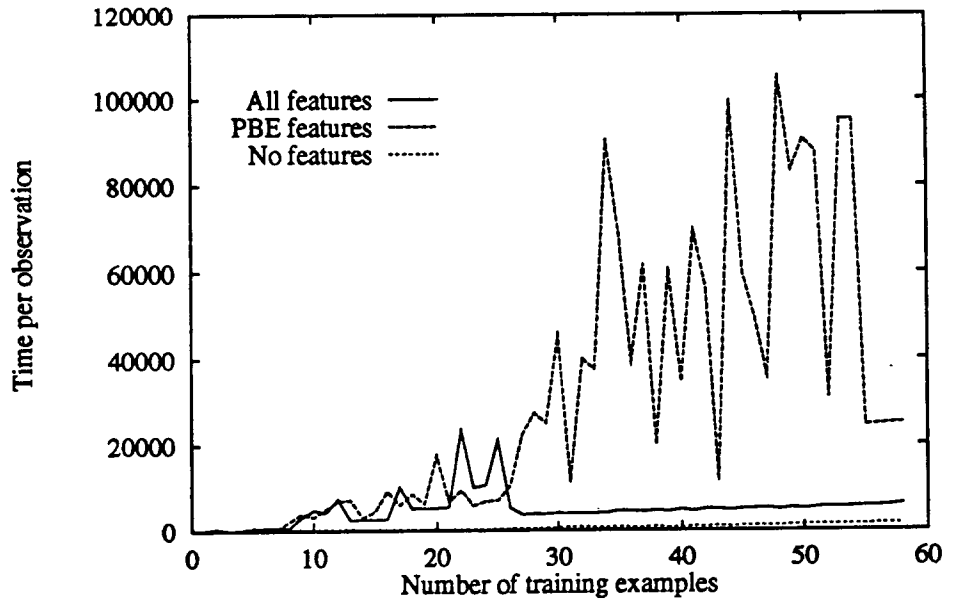

Figure 7.7: Bias tests in nasties world: time measurements

The timing results (Figure 7.7) are far more divergent—after 25 observations, all-features levels off, but PBE continues to grow, spiking erratically. Again, the spikes appear to correlate weakly with theory shifts: the time curve for all-features levels off at the same time as its accuracy curve flattens (t=24); PBE is continuously changing (and generating time spikes), but at the end both appear to be flattening out. The behavior of PBE in this test is inexplicable: although spikes appear in some of the other tests, none of them are as extreme.

Another interesting effect is that no-features starts off reasonably well, but quickly drops to barely 0.2 accuracy. In fact, what happened in the test was that PR got "trapped" by nasties. It had learned that being near nasties or performing a `:zap` action would cause its utility to drop, but not that zapping would cause a nasty to disappear. Because of this, it was unable to plan far enough ahead to recognize that the long-term benefit of zapping would outweigh the cost. Since it was continuously being bitten by nasties, it eventually formed a belief that no matter what it did, its utility would go down; this belief lowered its accuracy on the test set. Eventually, it chose `:zap` randomly (since 1/4 of the time it picks a random action instead of the apparent best action). When it did this—at $t = 40$—its accuracy started going back up.

Variability of results

To further explore the variability between test runs due to differing initial world states and random behavior of PR and the nasties, two tests using the same bias (`food-smell`, `nasty-smell`, `vision`, and `action`) were run in the nasties world, and tested on the same test set. The results are shown in Figure 7.8. In the first run, PR quickly found a fairly good theory, but later discarded it; its final accuracy was .58. In the second run, the effect was similar—an initial good theory discarded for poorer ones—but more dramatic. The good theory was discarded almost immediately for a much worse theory, and accuracy never got above .45.

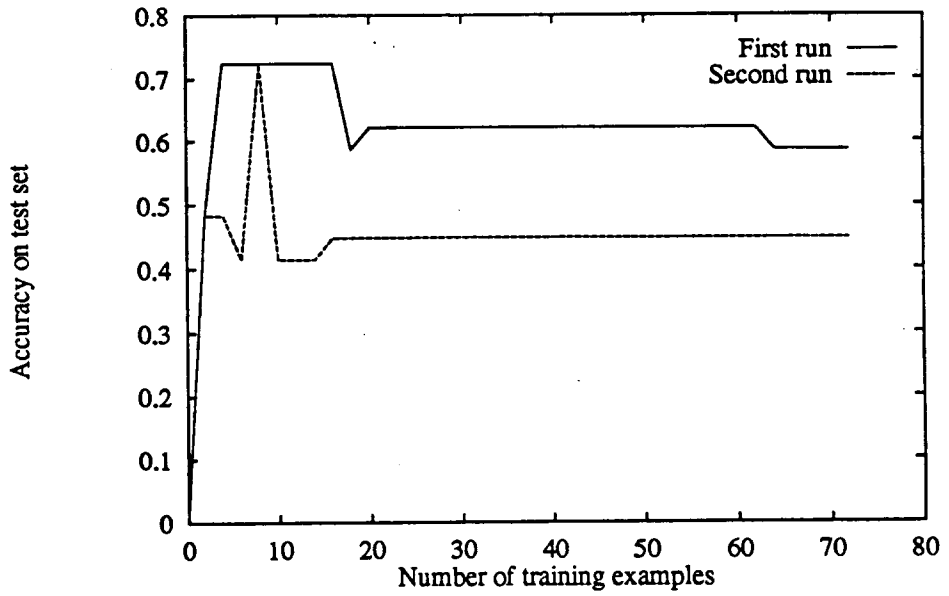

Figure 7.8: Variability of PR performance in the nasties world

Results

Since neither of the RALPH domains have a large number of features, and most of the features are relevant to predicting utility, the savings that can be gained by using PBE to select a subset of features for learning is not obvious here. The tests in Chapter 5 show the savings more clearly.

The need for better incremental learning algorithms is obvious from these results: all of the timing curves climb steadily as the number of observations increases. This is because all of the observations are stored and reprocessed when new theories are generated.

PR is developing good predictive theories, though: this is shown by the fact that PBE and all-features consistently perform significantly better than no-features (a single default rule).

7.2.2 Testing Priors

In these tests, the four prior probability distributions described in Chapter 6 (uniform distribution of theories, rule-level classification, feature-level classification, and Huffman encoding of features) were compared to each other. The four priors were run in the small-food world, and then on two separate runs in the nasties world.

Within all of the distributions, conflicts between theories with equal probability were resolved by choosing the shorter one (i.e., the one with fewer symbols). In both the feature-level distribution and the Huffman encoding, actions were considered to be "free" (i.e., they were not counted towards k, the number of features, and are taken to have length 0 in the Huffman code). This represents a bias towards describing outcomes based on the agent's actions.

The stronger biases towards simplicity—Huffman encoding and the feature-level distribution—are expected to perform better on the test set in more complex worlds. The reason for this expectation is that in a complex world, the training set is not expected to be representative of the entire world, especially initially, when the agent has not collected many observations. In this case, the agent should try to generalize its experiences, rather than forming highly specialized theories that precisely describe its training data. On the other hand, in simpler domains, a stronger bias towards simplicity will prevent the agent from extracting important dependencies.

The theories learned under the weaker biases will tend to be larger (with more rules and features), so we would expect that the time spent learning (processing observations into the theories and searching for neighboring theories) would be greater with these biases.

Small-food world

The small-food world has a fairly simple, deterministic theory for predicting Δu. Not surprisingly, all of the priors yielded approximately equal results in this domain.

The accuracy results can be seen in Figure 7.9. Since there is a deterministic theory, explaining everything that happens to the agent is acceptable, so the weak

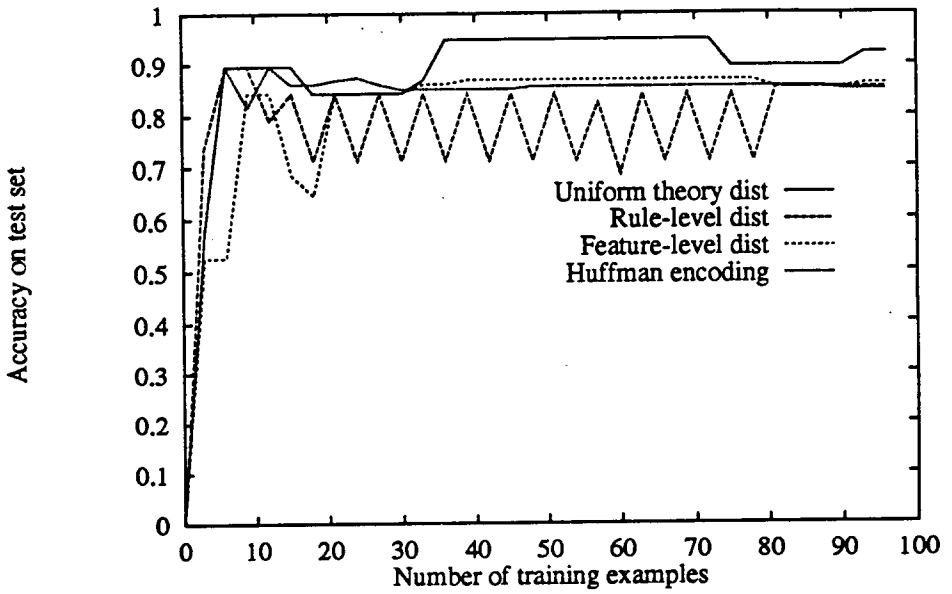

Figure 7.9: Prior-probability tests in small-food world: accuracy measurements

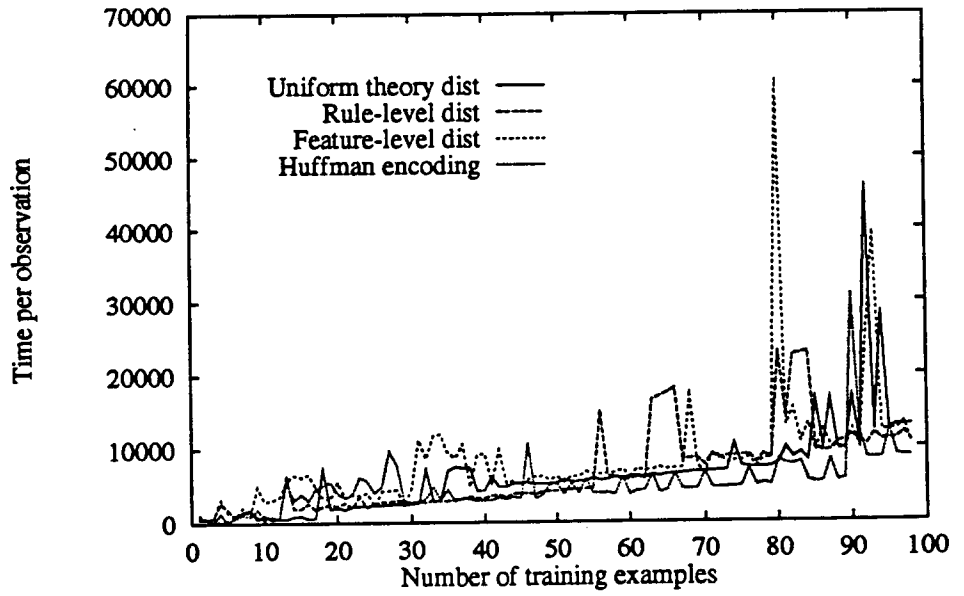

Figure 7.10: Prior-probability tests in small-food world: time measurements

biases perform well. Conversely, there exist fairly simple, correct theories, so the strong biases also succeed.

The uniform-rule distribution exhibits an odd oscillation. The underlying cause for this is that the agent has found two theories that appear nearly equal, and is shifting back and forth between the two. Why it shifts so regularly is not clear, but this phenomenon has been observed in other tests.

No significant differences in timing (Figure 7.10) are apparent: the feature-level distribution has a particularly noticeable spike, matching a theory shift that can be seen in Figure 7.9, but does not otherwise show a pattern of higher cost.

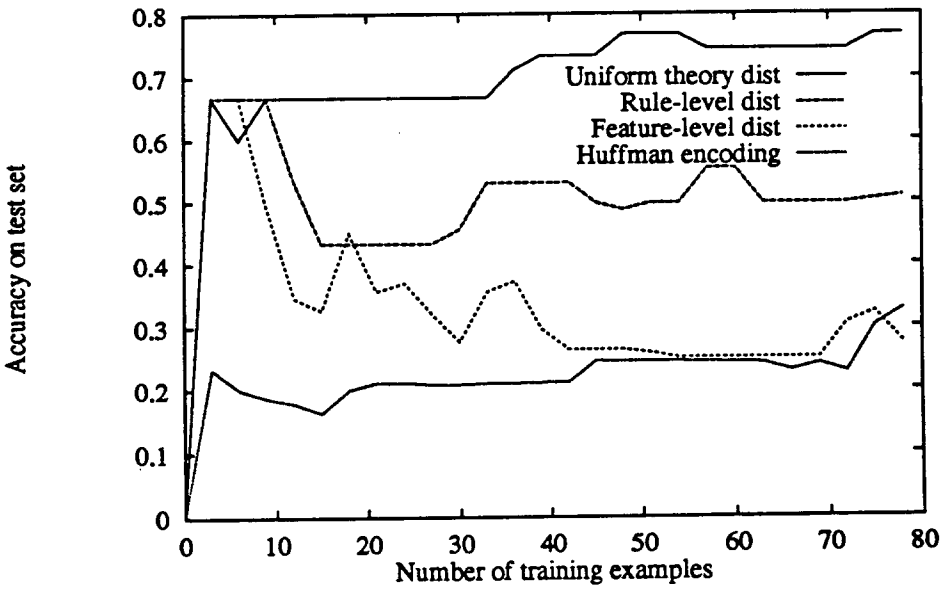

Figure 7.11: Prior-probability tests in nasties world (set A): accuracy measurements

Nasties world A

The results of comparing the four priors in the nasties world were more surprising. In this more complex world, we expected that the stronger biases would perform better, by not explaining noise. In fact, the results in Figure 7.11 show exactly the opposite: the uniform distribution on theories clearly outperformed the

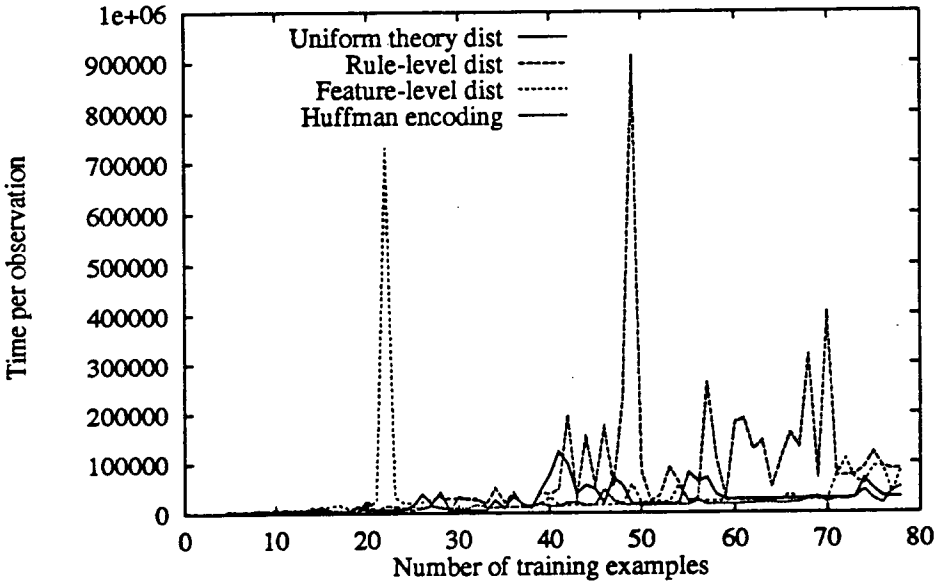

Figure 7.12: Prior-probability tests in nasties world (set A): time measurements

other priors. In general, the stronger the bias, the lower accuracy it achieved. At the end of the run, the accuracy of the Huffman encoding does appear to be increasing slightly, but the test ended before we could tell if this trend would continue.

The explanation for these results is that the domain is more complex than the small-food world (so that there is no theory that is both very simple and deterministic), but not complex enough for the strong biases to be effective. Because the correct theory is not simple, the strong biases fail to find it; because the world is not highly random, the explanations formed by the weaker biases are reasonable.

Nasties world B

To make sure that the results in the previous test were not simply due to coincidence, we ran a second set of tests in the nasties world. The results still do not show a preference for the stronger biases, but these biases are doing better than in the first test. The feature-level distribution and Huffman encoding still are not performing well. The uniform-feature distribution test ran into the same problem

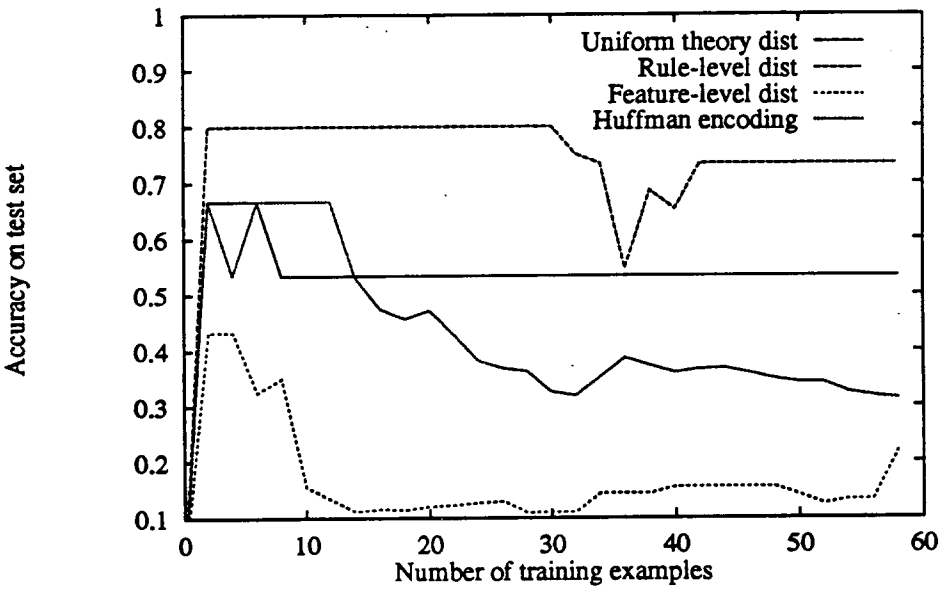

Figure 7.13: Prior-probability tests in nasties world (set B): accuracy measurements

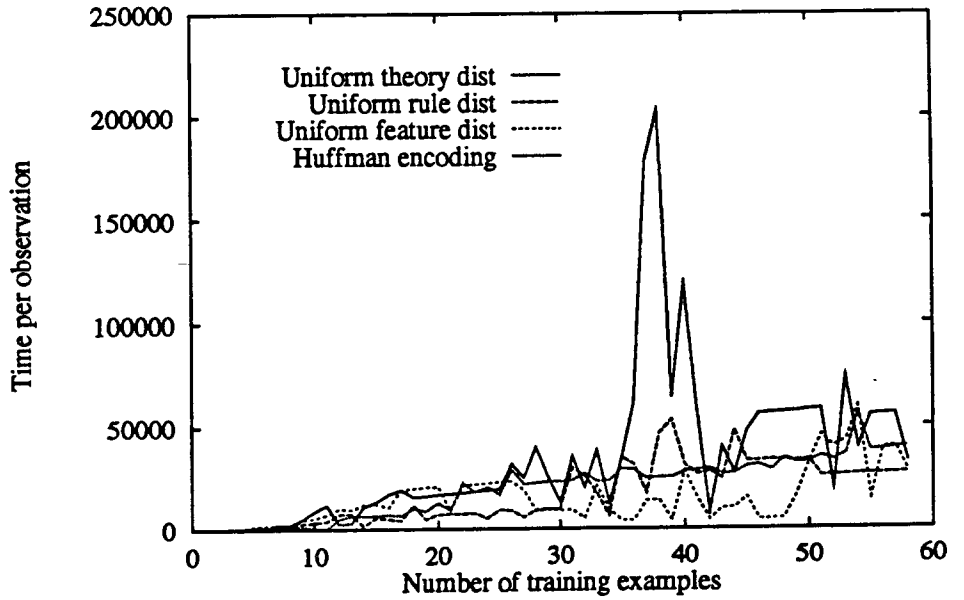

Figure 7.14: Prior-probability tests in nasties world (set B): time measurements

Distribution	Training set	Test set
Uniform theory	.84	.53
Rule-level	.92	.74
Feature-level	.71	.31
Huffman code	.49	.21

Table 7.1: Final accuracy in nasties test B

as no-features in the bias tests in the nasties world: it was trapped by nasties and spent a lot of time collecting observations of being bitten. The rule-level distribution, however, performed noticeably better than the uniform distribution on theories.

On this test, we also measured the accuracy on the training set. In a highly complex world, a set of observations of the size we collected would not be expected to be very representative of the world. In this case, accuracy on the training set and accuracy on the test set would not correlate well (hence the need for a simplicity bias to avoid explaining the random deviations in the training set). However, although the agent consistently made better predictions on the training set than on the test set, the accuracies do correlate: as Table 7.1 shows, the rule-level distribution has the highest accuracy on both the test set and the training set, followed by the uniform distribution on theories, the feature-level distribution, and Huffman encoding.

Results

In the small-food world, the uniform distribution on theories learns best; the rule-level distribution is not far behind. In the nasties world, the feature-level distribution and Huffman encoding appear to be overly strong; the rule-level distribution did better than the uniform distribution on theories in one test, but not as well in the other. In still more complex worlds, with a higher degree of randomness, the need for a simplicity bias may become more apparent. In this case, the agent should use whatever knowledge is available about the complexity of the world to determine which prior distribution to use.

7.2.3 Testing Learning Goals

In this section, we show the values that PAGODA generates for learning goals, given a learned theory, in the small-food and nasties worlds. We also show the improved performance resulting from adding the best learning goal to the system in the small-food world.

We expect to see a high correlation between intuitive usefulness of goals and their assigned values. When an additional learning goal is added, the agent should take more time to process each observation, since two theories are being formed, but have higher utility due to its improved ability to plan.

Small-food world goal values

The best theory generated by the PBE bias in the small-food world in this particular set of tests was:

$$
\begin{aligned}
&R_1(n=66)\ \square \quad \rightarrow_{.1.0} \quad \Delta\mathrm{u}(t+1,-10)\\
&R_2(n=9)\ \texttt{food-smell}(t,20) \wedge \texttt{action}(t,\texttt{:munch}) \quad \rightarrow_{1.0} \quad \Delta\mathrm{u}(t+1,90)\\
&R_3(n=4)\ \texttt{food-smell}(t,5) \wedge \texttt{action}(t,\texttt{:move-forward}) \quad \rightarrow_{.75} \quad \Delta\mathrm{u}(t+1,-11)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \rightarrow_{.25} \quad \Delta\mathrm{u}(t+1,-10)
\end{aligned}
$$

Since `food-smell` is the only feature that appears in any of the rules, it is the only goal with non-zero value. The substitution sets for the initial learning goals (Δu and `action`) are:

$$
\begin{aligned}
S_1 &= \{R_1\}\\
S_2 &= \{R_2, R_1\}\\
S_3 &= \{R_3, R_1\}
\end{aligned}
$$

R_3's past instances are assigned to R_1's hypothetical instances, resulting in the final weights

$$
\begin{aligned}
m_1 &= 70\\
m_2 &= 9\\
m_3 &= 0
\end{aligned}
$$

The expected utility of a single-step plan in the initial world model is $(70 * (-10)) + (9 * 90)$, or 110. Adding the learning goal `food-smell` gives the substitution sets

$$\begin{aligned} S_1 &= \{R_1, R_2, R_3\} \\ SE_2 &= \{R_2, R_1\} \\ S_3 &= \{R_3, R_1\} \end{aligned}$$

and the final weights

$$\begin{aligned} m_1 &= 0 \\ m_2 &= 79 \\ m_3 &= 0 \end{aligned}$$

so the the expected utility of the augmented world model is $79 * 90$, or 7110. The value of `food-smell` is $7110 - 110$, or 7000.

Small-food world behavior

PR was run using Δu as the only learning goal, and then using both Δu and `food-smell`. The results are shown in Figures 7.16 and 7.15. As expected, learning with two learning goals takes significantly more time to process each observation, but the cumulative utility is higher.

In the case where PR learns theories for both Δu and `food-smell`, it recognizes situations when it can get to food in one step (i.e., when it is next to and facing food). In this case, it moves to the food and eats it. However, if it is next to but not facing the food, it can't tell which way to turn; if it is not next to food, it can't plan far enough ahead to get to and eat the food. The planning ability is definitely an improvement over Δu alone (when PR simply wanders randomly until it happens to land on the food), but not as much as one might imagine, highlighting the difficulty of building a completely autonomous agent that can behave intelligently in an unfamiliar environment.

Nasties world goal values

The theory given in Figure 7.17 was generated using the PBE bias and rule-level prior probability distribution in the nasties domain, in the second set of tests

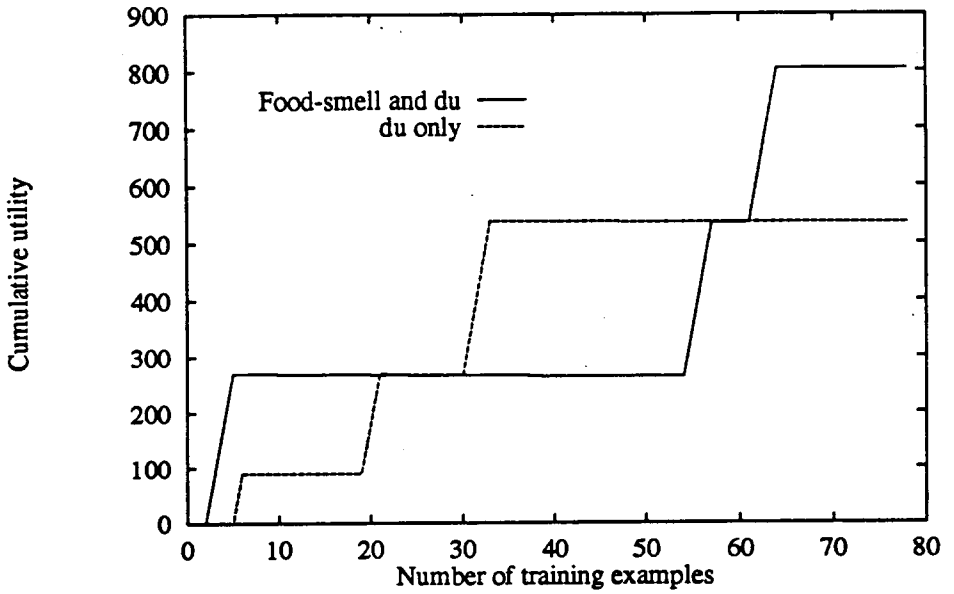

Figure 7.15: Goal tests in small food world: utility measurements

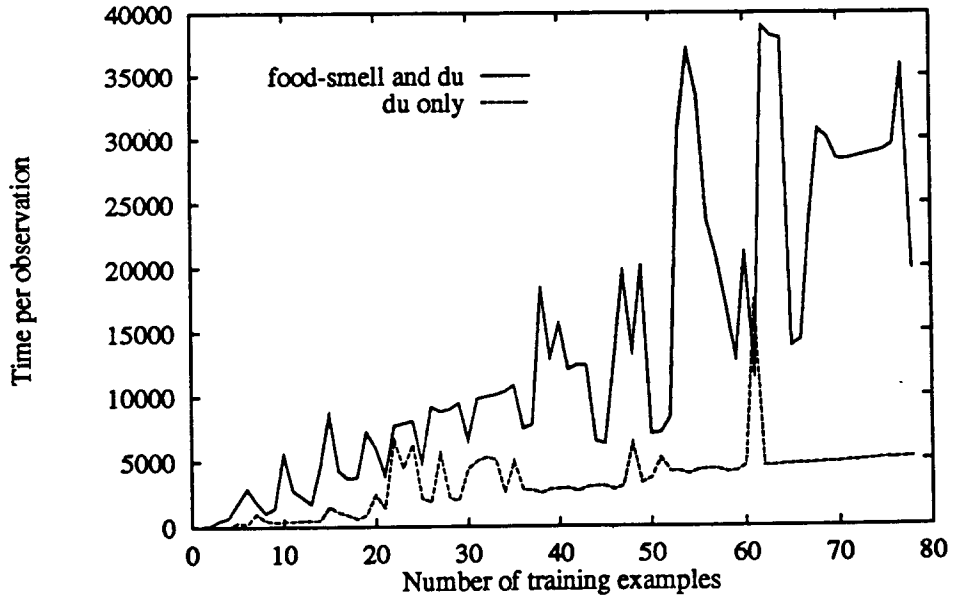

Figure 7.16: Goal tests in small food world: time measurements

described above.

R_1 : $\Box \rightarrow_{1.0} \Delta u(t+1, -10)))$

R_2 : nasty-smell$(t, 10) \rightarrow_{.29} \Delta u(t+1, -10)$
$\rightarrow_{.57} \Delta u(t+1, -60)$
$\rightarrow_{.14} \Delta u(t+1, -210)$

R_3 : nasty-smell$(t, 10) \wedge$ vision$(t,$ inanimate-object$, 2) \rightarrow_{1.0} \Delta u(t+1, -60)$

R_4 : nasty-smell$(t, 20) \rightarrow_{.5} \Delta u(t+1, 40)$
$\rightarrow_{.5} \Delta u(t+1, -10)$

R_5 : food-smell$(t, 11) \wedge$ nasty-smell$(t, 20) \rightarrow_{1.0} \Delta u(t+1, -110)$

R_6 : action$(t,$:zap$) \wedge$ food-smell$(t, 11) \wedge$ nasty-smell$(t, 20)$
$\rightarrow_{1.0} \Delta u(t+1, -260)$

R_7 : nasty-smell$(t, 15) \rightarrow_{1.0} \Delta u(t+1, 90)$

R_8 : nasty-smell$(t, 11) \rightarrow_{1.0} \Delta u(t+1, -60)$

R_9 : food-smell$(t, 9) \wedge$ nasty-smell$(t, 11) \rightarrow_{1.0} \Delta u(t+1, -10)$

R_{10} : vision$(t,$ wall$, 1) \rightarrow_{.25} \Delta u(t+1, -11)$
$\rightarrow_{.75} \Delta u(t+1, -60)$

Figure 7.17: Theory learned in the nasties world

Table 7.2 shows the weight (n_r) and expected utility (EU_r) for each rule, and gives the substitution sets (S_r, a list of rule numbers), final weights (m_r), and overall expected utilities for the initial plan space and for the three candidate learning goals (nasty-smell, food-smell, and vision). Intuitively, we expect nasty-smell to have the highest value, because it appears most frequently.

The value of each potential learning goal is the expected utility of the plan space formed using the augmented world model minus the expected utility of the

Rule	n_r	EU_r	Initial plan space		nasty-smell		food-smell		vision	
			S_r	m_r	S_r	m_r	S_r	m_r	S_r	m_r
R_1	28	-10	1	28	1,2,4,7,8	4	1	28	1	35
R_2	7	-66.5	2	7	1,2,4,7,8	–	2	7	2	7
R_3	3	-60	3	3	1,3,4,7,8	–	2	7	2	7
R_4	2	15	4	2	1,2,4,7,8	–	4	6	4	2
R_5	2	-110	5	3	1,2,4,5,7,8	–	1,4,5	–	5	3
R_6	1	-260	6	1	1,2,4,5,6,7,8	–	1,4,5,6	–	6	1
R_7	1	90	7	1	1,2,4,7,8	57	7	1	7	1
R_8	11	-60	8	11	1,2,4,7,8	–	8	11	8	11
R_9	1	-10	9	1	1,2,4,7,8,9	–	1,8,9	1	9	1
R_{10}	4	-47.75	10	4	1,10	–	10	4	1,10	–
Expected utility			-37		83.4		-26.3		-32.1	

Table 7.2: Plan space utility for learning goals

initial plan space:

$$V(\texttt{nasty-smell}) = 83.4 - (-37) = 120.4$$
$$V(\texttt{food-smell}) = -26.3 - (-37) = 10.7$$
$$V(\texttt{vision}) = -32.1 - (-37) = 4.9$$

As expected, `nasty-smell` has a high value: if the agent could always choose a plan to predict its value, it would always choose the value 15, leading to $\Delta u = 90$, (according to its theory). `food-smell` has a slightly higher value than `vision`; this is because being able to determine values for the former would allow the agent to avoid applying rules R_5 and R_6, both with large negative utility.

Results

The benefit (in terms of added utility) of adding high-value learning goals can be seen in the tests shown in this section. The relative values of the goals match well with intuitions about which goals are useful, given the agent's theory. Of course, an incorrect theory will lead to bad decisions from an omniscient observer's point of view, but the decisions are still rational for the agent.

The current goal values only represent the utility gained by learning them, and does not include the associated costs of learning and planning with an additional

theory. In order for the agent to decide automatically when it is worth adding a goal, a model of these costs is needed.

7.3 Conclusions

The tests clearly show that PR is learning: the predictions made by its learned theories are significantly better than chance. However, the results do not show perfect accuracy, due to the complexity of the domain as well as limitations of the agent.

The focus of the learning mechanism in PAGODA was on the probabilistic representation and evaluation of theories, and not on the heuristic search for theories. The search procedure used by PAGODA works well enough that good theories can be found, as shown in the tests, but needs a sounder theoretical foundation and a more efficient implementation. The variability of the timing results is most likely due to anomalies in the search which have not been analyzed or measured; this is supported by the apparent correlation between time spikes and theory shift.

An ideal heuristic search procedure would be not only theoretically sound (provably correct and efficient) but incremental: that is, it would not store all previous instances to be reprocessed when new observations arrive. Incremental learning algorithms are necessary to avoid the steadily increasing costs of processing new observations that are evident in the test results.

Another open area for future research is planning using the learned probabilistic theories. The simple forward-chaining search without pruning that PAGODA currently uses is too simplistic and cumbersome to allow the agent to make good decisions. Opportunities for improvements to the planning mechanism are discussed in Chapter 9.

The most important conclusion to be drawn from these tests is that autonomous intelligent agents such as PAGODA are very complex systems, in which it is difficult to isolate and measure the effects and behavior of the individual components. However, we believe that the tight conceptual integration of the system is essential, and that only by continuing to develop all of the components in parallel can the

performance be improved.

Chapter 8

Related Work

The goal of the research described in this thesis is to build an autonomous, resource-bounded learning agent which can function in a variety of environments. There are many issues involved in intelligent agent design, and the interrelations between them are complex. Most existing research addresses only a few of these issues, and tends to ignore the relations between them. In particular, very little work has been done on the problem of autonomous learning under uncertainty.

In the next section, we present a system of axes along which machine learning research can be classified, discuss where an ideal system would lie, and classify PAGODA in terms of the axes. Section 8.2 gives some background on the problem of inductive learning, including the philosophy of induction and early machine learning work. Section 8.3 presents previous work on autonomous learning and related problems. Section 8.4 presents a variety of approaches to the problem of defining and changing concept learning representations. Finally, Section 8.5 discusses existing work on probabilistic planning.

Related work on learning under uncertainty was discussed in Chapter 2.

8.1 Classification of Machine Learning Research

We have developed a set of axes (Figure 8.1) along which machine learning research can be classified. The axes are broken down into four subgroups: character-

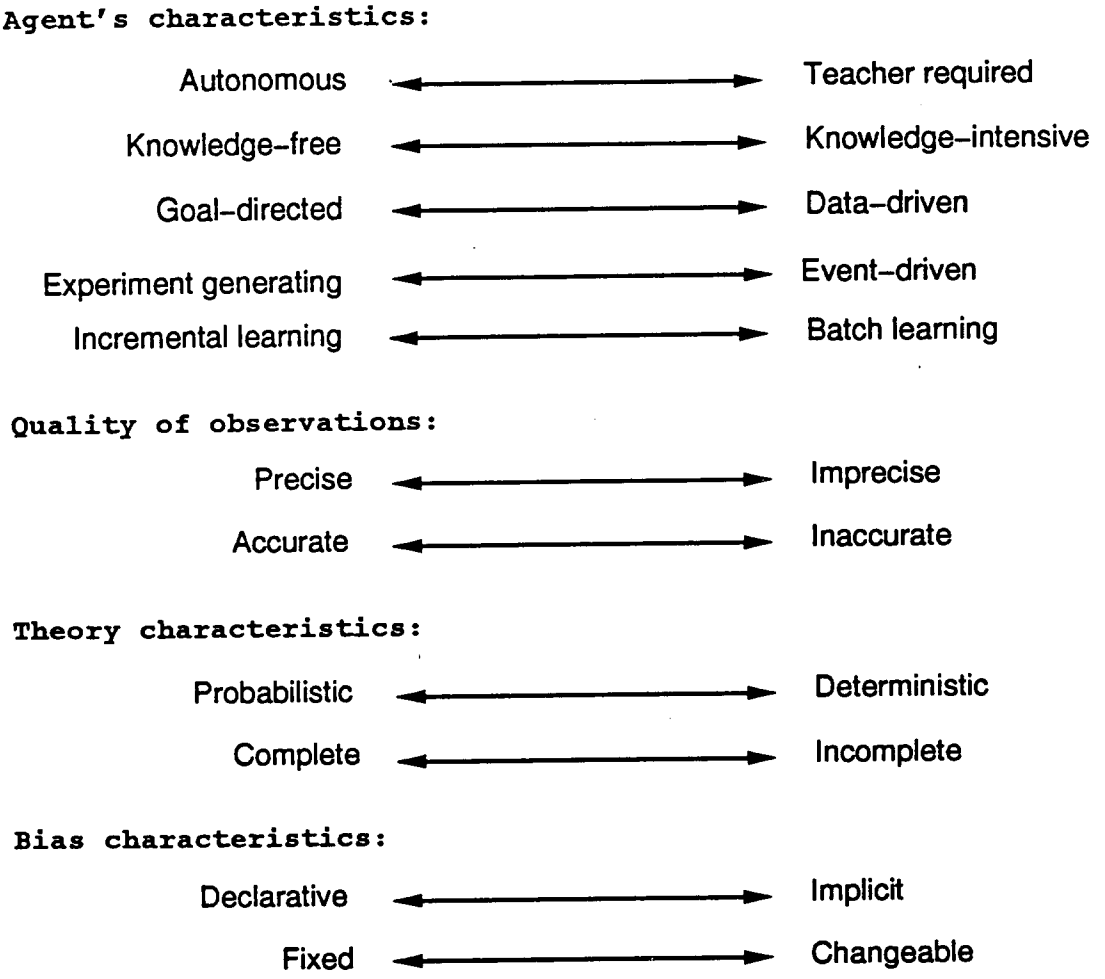

Figure 8.1: Axes for Classifying Machine Learning Systems

istics of the agent's learning system, quality of the observations, characteristics of the learned theory, and characteristics of the learning bias.

Agent's characteristics: The agent may be characterized by the degree of supervision required, the amount of background knowledge used in learning, how the agent decides what to learn, whether the agent actively experiments on its environment, and whether the agent learns incrementally or all-at-once.

Quality of observations: The observations may be more or less precise (i.e., grain size may vary) and more or less accurate (i.e., degree of noise may vary). These

attributes will depend on the agent's sensors.

Theory characteristics: The theories learned may be nondeterministic (in the sense that they explicitly represent uncertainty about the world) or deterministic, and complete (i.e., make a prediction in all situations) or incomplete (occasionally answer "I don't know").

Bias characteristics: The input language and hypothesis language will vary; they may be stated declaratively or be implicit in the learning mechanism, and may be fixed or changeable.

8.1.1 Ideal Autonomous System

The characteristics of the ideal autonomous agent vary along the axes, depending on the current environment. It operates autonomously, but takes advice from a teacher when available; operates without any domain-specific knowledge, but uses a domain theory if it has one; is goal-driven but has enough curiosity to notice significant regularities in the data; performs experiments if there is time and the risk factor is low enough, but simply *acts* when necessary; and performs incremental learning to the degree that its limited time and memory resources require.

Obviously, an agent's behavior and performance will be affected by the quality of its sensors, but an ideal agent should be able to function as well as possible regardless of the degree of precision or accuracy of its sensors.

Theories need to express uncertainty in order to be fully general, and an agent should always be able to make *some* (possibly probabilistic) prediction, but should also have enough meta-knowledge about the information used to learn its predictive rules, and about its information-gathering processes, that it can reason explicitly about the accuracy of its predictions in order to decide whether to trust them.

Finally, we believe that background knowledge about the world (which can itself be learned) should be used to find a declarative learning bias, and that this bias must be modifiable by the agent in order for learning to be effective in a complex

domain.

In other words, an ideal agent exhibits limited rationality. If its background knowledge is represented as fully and explicitly as possible, the agent can more effectively determine what it really knows, where the gaps in its knowledge are, and how it should guide its learning and planning behavior to behave optimally with bounded resources.

8.1.2 Classification of PAGODA

PAGODA, the agent model described in this thesis, comes closer than any other single system we know of to meeting these criteria for an ideal agent. It behaves largely autonomously, but can accept external input in the form of background knowledge which is used to select a learning bias. The more precise the background knowledge is, the stronger its learning biases are. Because of this, it performs well in both knowledge-free and knowledge-rich environments.

Because PAGODA does not incorporate a model of reasoning with limited resources, it does not modify its learning behavior in a fully general way: it is always goal-directed; it does not do experiment generation; and its learning is not completely incremental. However, because of the modularity of PAGODA's architecture, a more intelligent planner and a better model incremental learning could be added to PAGODA, achieving generality along these axes.

PAGODA does not make any assumptions about the content or quality of its sensors—although of course its learning capacity will naturally be limited by inadequate sensors. Because the theories it learns are probabilistic, noisy sensor data or an inability to distinguish different world states simply cause PAGODA to build a less accurate theory. The effects of sensor inaccuracies and limitations were discussed in Chapter 6.

The theories learned by PAGODA are complete probabilistic theories, but it does not reason about the quality of its theories (that is, it does not use second order probabilities to reason about how good its predictions are expected to be). In order to do this, extensions to the probabilistic evaluation techniques and to the planner

will be needed. Finally, PAGODA's learning bias is declarative and changeable, as desired. The technique for probabilistically evaluating bias can be extended to use other forms of background knowledge in selecting a bias.

In the remaining sections, we will indicate the strengths and weaknesses of research related to this work with respect to our learning classification of the "ideal" system.

8.2 Inductive Learning

The problem of induction has two parts: first, how can an agent reason from observations to predictive rules or to specific predictions; and second, how can this reasoning process be justified?

Hume's skeptical view of induction [Hume, 1975] was that induction is not rational, and so the second question has no answer. Goodman's "new riddle of induction" [1983] shifted the focus from the latter question to the former: rather than attempting to justify induction (which, he argues, cannot be done) we should define precisely what is meant by a valid inductive inference. Goodman concludes that the problem can be reduced to defining which regularities we are willing to consider in forming inductive hypotheses. Specifically, which of our predicates, in a given inductive context, are *projectible*; that is, which properties of past objects may be projected onto future similar objects?

In the field of computational learning theory, analysis of induction has traditionally been based on Gold's theory of inductive inference [1967]. His definition of the problem is as follows: an agent is provided with a sequence of examples that are classified as positive or negative instances of some concept. After each instance is presented, the agent must "guess" what the definition of the concept is. A concept class is *learnable* if there exists an algorithm that the agent may use such that, after some finite time, all of the agent's guesses will be correct (that is, the algorithm converges to the correct concept in a finite number of instances). This paradigm is known as *learning in the limit.*

More recently, Valiant [1984] expanded this analysis to allow probabilis-

tic convergence and to provide a better model of the computational complexity of learning. A concept class is *PAC-learnable* (PAC stands for "probably approximately correct") if there exists an algorithm that, with probability $1 - \delta$, finds a concept that has error less than or equal to ϵ, using a number of examples that is polynomial in $1/\epsilon$ and δ. Later research has extended this model to analyze the effects of noise [Angluin and Laird, 1986, Kearns and Li, 1987] and to allow arbitrary cost functions [Haussler, 1989].

The machine learning community has generally viewed induction as a problem of searching a space of potential hypotheses to find a consistent one. Michalski's description of the *Star* system [1983] gives a large set of inference rules, which can be thought of as operators for searching the space. These operators include both selective generalization rules (e.g., dropping a condition or climbing a generalization tree) and constructive induction rules (e.g., forming new terms by counting arguments or by generating chain properties).

Muggleton [1988] used the principle of inverse resolution as the basis for inductive inference in CIGOL. Induction is modeled as a complementary technique to deduction, involving finding a theory that would deductively predict the data. The theory is found using operators that are inversions of the resolution steps of logical deductive inference.

One of the limitations of all of the systems presented here, and indeed of most machine learning systems, is that they only learn deterministic theories. Even the PAC model is intended to analyze learnability where the true concepts are deterministic. In Chapter 2, we presented some approaches from probability theory and machine learning that can handle nondeterministic and noisy environments. Those systems, though, represent only a small fraction of machine learning research to date.

8.3 Autonomous Learning

Existing work on autonomous learning falls primarily into two categories: discovery systems, which do concept learning without requiring a teacher to classify instances, and conceptual clustering techniques, which create classifications ("clus-

ters") for unsorted data. We do not survey conceptual clustering techniques, since they are not intended to be used for predictive learning.

Research on discovery has focused on two tasks: scientific theory formation and exploring a robot world. Research on the former includes AM, Eurisko and the BACON systems. Research on the latter includes Kuipers' Map-Learning Critter, Rivest and Schapire's work on learning DFAs, and Mitchell et al.'s research on robot manipulation.

Scientific Theory Formation

A number of AI systems have been built to perform scientific discovery in a variety of domains. These systems generally incorporate hand-tailored domain knowledge and heuristics for generating theories in the domain.

AM [Lenat, 1979] discovers mathematical and set-theoretic concepts. It uses an "interestingness" heuristic to evaluate new concepts, which are found using heuristic production rules. AM is essentially a best-first search with carefully tailored search operators (the heuristic production rules) and evaluation function (the interestingness heuristic).

EURISKO [Lenat, 1982a, Lenat, 1982b, Lenat and Brown, 1984] is an extension of AM that adds a heuristic-description language, allowing the system to be applied to new domains and to find new heuristics using a meta-discovery process. Domain-specific information is still required, but the meta-rules for finding new heuristics are somewhat more general. Still, the heuristics are *ad hoc*, with no formal justification, and the system must be fine-tuned by hand.

BACON [Langley *et al.*, 1983, Langley *et al.*, 1986] discovers empirical laws of chemistry by incorporating heuristics which examine experimental data to find regularities. Some fairly general heuristics are used, but the system does not have methods for evaluating and comparing multiple theories.

None of these systems have a theoretically justified method for choosing among competing theories. Perhaps more importantly, none directly address the question of scientific bias. Kuhn [1962] argues convincingly that existing theories and

paradigms provide the basis for scientific experimentation and theory formation, by providing an initial bias and guiding the search for questions to ask. In order for automated systems to use these biases, appropriate background knowledge must be identified and incorporated in their design. However, the system must be capable of overriding its initial beliefs and forming novel theories, or it will never discover anything new.

Exploring Unfamiliar Environments

The Map-Learning Critter (MLC) [Kuipers, 1985] learns the structure of a simulated environment, and interprets its own actions and senses as they affect each other via the environment. The MLC embodies a strong *a priori* hypothesis that the environment is a large space made of places connected by paths. The learning process, therefore, consists of constructing a cognitive map of the environment. It does this by classifying actions as "turn-like," "travel-like," or "other," finding inverses, and exploring the environment. The exploration strategy consists of determining the current place, exploring the current path, and exploring the network randomly.

The MLC does not work well in complex environments or environments that are not "map-like" (i.e., that contain objects that can be manipulated or states that can be affected without moving). "Other" actions are simply ignored. Also, many assumptions and definitions are built in; these place a strong constraint on the types of worlds that can be learned, and therefore make learning easier in these particular worlds, but impossible in more general worlds.

A method for learning the exact structure of a deterministic finite-state environment is described in [Rivest and Schapire, 1987]. A perfect model consists of a set of *canonical tests* (sequences of actions leading to a predicted sensation) and the values of the canonical tests in the current state; the inference procedure learns these canonical tests. W, a set of tests to be examined, is initialized to contain the set of sensations, and the set of canonical tests is initialized to be empty. The least (shortest) test t is removed from W and is tested for equivalence to a known canonical test. Equivalence is determined by repeatedly executing the new test until

the outcome becomes periodic; randomization is used to increase the confidence that true periodicity has been found. If t's outcome is not then found to be equivalent to some existing canonical test, it is added to the set of canonical tests, and for each action a, a new test at is added to W. This process is repeated until W is empty. Because of the need for the tests to become periodic when executed repeatedly, the probable correctness of the method can be proved in environments where the effects of actions can be reversed, and in permutation environments in general. However, its efficacy in other types of environments is not known.

Mitchell's recent work on robot manipulation [Mitchell *et al.*, 1989] uses a variation of explanation-based learning (EBL) driven by an incomplete domain theory. Uncertainty derives from a limited ability to observe the world and from complexity of computations. The planning mechanism constructs a strong plan (one that succeeds for all possible outcomes) if one can be found; if not, a weak plan (one that succeeds for some possible outcome) is constructed using the incomplete domain theory. Execution monitoring is used to stop the plan if and when failure is observed, construct a plausible explanation for the failure (based on general physics knowledge), and infer general conditions that would cause or avoid the error. When the goal is achieved, the domain theory is updated to reflect the success of the final plan.

Both the MLC and Rivest and Schapire's method work only in deterministic environments. Mitchell's work deals with uncertainty in the sense that the plausible theories generated are not necessarily correct, but since it does not represent uncertainty explicitly, it is not clear how well it would deal with highly complex environments or environments with true randomness or noise.

8.4 Bias

Mitchell [1980] showed that in order for learning to take place, the system must have some basis for choosing one consistent theory over another; that is, the search space must be *biased*, either because it contains only a subset of the possible theories, or by use of a preference function. In many cases, this bias can be represented in the language used for learning.

Rosch's work on basic concepts [1976] and Goodman's theory of projectibility [1955] emphasize the effects that language has on what hypotheses we form (and, conversely, the effects that learning has on the language we use). Rosch showed that there are certain levels of descriptions that are more salient than others for purposes of classification (e.g., "chair" is a basic concept, and is more likely to be used to describe a new concept than either "furniture" or "rocking chair"). Goodman's theory states that predicates that have been used in the past to form theories should be more likely to be used in the future; these *projectible* predicates reinforce themselves.

Research on representing bias declaratively is presented in the next section; Section 8.4.2 discusses approaches to shifting bias.

8.4.1 Declarative Bias

Russell and Grosof [1987] use *determinations* to describe a concept language bias declaratively. A determination provides a sufficient set of features for learning a particular concept. If determinations are chained together, the chaining process can be used to find a *tree-structured bias*, where the interaction of the features is constrained by the tree structure [Russell, 1988].

Using determinations to represent bias ignores the questions of what to do if the background knowledge is incomplete or incorrect, so that the agent finds it impossible to learn a good concept definition, and of how to express a preference ordering on possible languages.

One way to extend the determination approach is to include operationality information about predicates, and find the best (most operational) bias [Getoor, 1989]. Another approach is to use a probabilistic version of determinations. Russell [1986] discusses probabilistic forms of determinations, but not how to use them for this purpose. The work described in this thesis on probabilistically evaluating biases extends the concept of declarative bias to probabilistic domains.

Martin and Billman's variability bias [1991] is similar to the uniformities used by PAGODA, but captures more information about the expected distribution of outcomes than uniformities do. However, it is not clear how to form variability

biases automatically. Also, the variability biases are used to weight expectations for predictions, rather than as a tool for selecting a bias. In other words, a single variability bias is used both to determine which features to examine and to weight the predicted distribution according to the expectations provided by the variability bias.

8.4.2 Shift of Bias

When a learning system's initial bias—whether provided by the designer or selected by the system automatically—is determined to be incorrect, the agent should be able to select a new bias. Solving this problem requires determining when the correct bias is inadequate, finding alternative biases, possibly evaluating competing biases, and finding a theory using the new bias.

Utgoff's STABB [1986] is the earliest work that directly addresses this problem. STABB uses version-space collapse as a signal that the current bias is inadequate (since no consistent concept can be found). A search is then initiated for a new term to add to the feature hierarchy that captures a necessary distinction. When the term is found, the version space is recomputed from scratch using the new feature hierarchy. STABB assumes deterministic theories, is computationally expensive (since the version space must be recomputed every time a new term is added), and can only shift the bias in a very limited way.

Muggleton's Duce [1987] and CIGOL [1988] both form new terms as part of the inductive learning process. The terms learned represent disjunctions which allow more concise descriptions of the system's theories. Muggleton assumes deterministic theories, and requires an oracle to tell the system whether the new terms are worth adding.

STAGGER [1987a] adds new features—conjunctions and disjunctions of existing features—based on the sufficiency and necessity values of the existing features. This technique is basically *ad hoc*, and provides no formal consideration of the tradeoffs associated with adding new features. However, since the theory representation represents the influence of features independently, adding a new feature does not

require any recomputation of the existing theory.

A number of researchers have investigated the problem of constructive induction; that is, of generating new features to make learning more efficient or the representation more expressive. A session of the 1991 Machine Learning Workshop was devoted to research on this topic [Birnbaum and Collins, 1991].

8.5 Planning

Planning a course of action in a probabilistic domain is a relatively unexplored research area. "Classical" AI planning techniques require a complete, deterministic world model, where the outcome of applying an operator in a given situation is known with certainty. Planning in this case becomes a relatively straightforward problem of heuristic search. In domains containing uncertainty, the planning problem is more complex, requiring the agent to maximize its probability of success rather than finding a path that is necessarily guaranteed to reach a goal state. In addition, agents with bounded resources must balance the amount of time spent planning with the cost of deliberation.

Decision theory [von Neumann and Morgenstern, 1947, Savage, 1977] provides a formal theory of rational action which can be used to make decisions in uncertain domains. Decision theory uses information about probabilities and utilities of events to select optimal courses of action; computational limits can be taken into account by including models of the cost of delaying actions [Pollard, 1969].

AI researchers have recently begun to study the direct application of decision theory to intelligent real-time behavior. Protos [Horvitz *et al.*, 1989] is a decision-theoretic system for real-time control, which uses the expected value of computation to decide whether to compute further or whether to act. Smith [1987] has developed a decision-theoretic approach to controlling heuristic search which uses models of the costs and probabilities of achieving goals to control the search process. Russell and Wefald [1991] address the general problem of limited rationality at an abstract level, particularly the use of metareasoning to control deliberation.

In the following sections, we survey two other approaches to planning with

uncertainty: reactive strategies are discussed in Section 8.5.1 and deliberative planning with uncertain knowledge is discussed in Section 8.5.2.

8.5.1 Reactive Strategies

Reactive techniques address the problem of planning in complex, nondeterministic domains by providing an agent with a strategy (set of condition-action rules) that allows it to "react" quickly to any given situation [Brooks, 1991, Agre and Chapman, 1987]. The primary problem with reactive systems is that the strategies are difficult to build. Recently, research has begun to focus on learning reactive strategies.

Delayed reinforcement learning [Kaelbling, 1990, Sutton, 1990] is a method for learning to associate optimal (maximum-utility) actions with each possible situation. Utilities, or reinforcement values, are propagated backwards and associated with the actions that led to them. Eventually—given enough observations—the system will converge to the actual expected discounted utility of actions; these utilities provide the system with an optimal reactive strategy (in every situation, the system should choose the action with highest expected utility). Current research in this area does not adequately address the problem of generalizing these utility models (but see [Chapman and Kaelbling, 1990] for preliminary work on this problem). Also, the convergence of this method can be extremely slow, and convergence is only guaranteed if every possible situation is observed an unbounded number of times.

Dyna-Q [Sutton, 1990] uses a similar technique to learn a policy. The planning method is refined by using a Boltzmann distribution with annealing to choose actions: the probability of selecting an action depends on its expected value, and the likelihood of selecting the best action under this distribution increases as more evidence is collected. An exploration bonus, proportional to the time since an action was last tried in a particular situation, is included, allowing the agent to continually test its beliefs and thus cope with changing environments.

Robo-Soar [Laird and Rosenbloom, 1990] uses the Soar learning mechanism of "chunking" to generalize previously generated plans and store them for future

problem solving. These stored plans, which are similar to macro-operators, allow the system to solve similar problems more quickly. A set of stored plans is equivalent to a set of reactive rules for guiding future behavior.

8.5.2 Deliberative Planning with Uncertainty

A number of researchers have developed methods for intelligent systems to build plans using probabilistic world models by propagating error and uncertainty.

Brooks [1982] gives a symbolic method for propagating error bounds through a robot plan (sequence of motions and sensing operations with conditional branches). The errors include placement of objects, tolerances in the manufacturing of the objects, and positional error of the robot. Plans are refined by adding operations and changing preconditions and constraints until the resulting plan is determined to be feasible (i.e., to be guaranteed to succeed).

Lozano-Pérez [1984] defines a *compliant motion* as one that uses feedback (e.g., pressure readings from a robot arm) to reduce uncertainty and control the robot's motion. His method develops a robot plan using compliant actions which is guaranteed to reach a goal state from all possible initial states. The plan is generated by backward chaining from the goal state, propagating positional uncertainty.

Qi and Poole [1991] solve the problem of navigation under uncertainty by modeling distances between points as switches that have a specified probability of being open (so that the path between the points is impassable) and a cost if closed (i.e., the cost of traversing that path). They treat the model as a finite-state Markovian decision problem, and give a minimal-cost solution for systematically exploring the environment until arriving at the goal (if possible).

Temporal projection [Drummond and Bresina, 1990] is a planning method that attempts to maximize the probability of goal satisfaction by performing a focused beam search in the space of possible action outcomes. The goals are simple primitive states; the system does not attempt to maximize overall utility, or to resolve conflicting goals.

Kanazawa and Dean's method of probabilistic temporal reasoning [1989]

represents the world model as an influence diagram with an explicit temporal component. At compile time, this model is run through simulations to determine a fixed optimal time for deliberation (i.e., the amount of computation time that maximizes average expected utility in a real-time decision situation). This optimal time is used to solve decision problems at run time. The world model is fairly simplistic, though, and grows quickly in size as the temporal complexity grows. Additionally, since the optimal time is determined at compile time, continuous learning cannot be incorporated into the process.

Chapter 9

Future Work and Conclusions

PAGODA provides a model for building intelligent autonomous agents that learn and function in complex, uncertain environments. Methods for selecting learning tasks, representing probabilistic knowledge, selecting and changing learning bias, learning probabilistic theories, and planning with the learned theories are integrated into a single system.

PAGODA has been implemented and tested in a simulated robot domain (RALPH), and the model does allow effective learning in this domain. However, a number of open problems remain to be solved before PAGODA can be extended to more complex domains.

We present some of these problems in the next four sections, breaking them down into the same four research areas as before: Goal-Directed Learning, selecting a bias, probabilistic learning, and probabilistic planning. Finally, Section 9.5 summarizes the contributions of the thesis and presents our conclusions.

9.1 Goal-Directed Learning

The basic principle of Goal-Directed Learning (GDL) is that intelligent agents should learn theories that will maximize their average utility in the long run. This means constraining the set of features learned in a complex domain to those which, if predicted by the world model, would enable the most effective planning. GDL does

this by selecting the features that will enable high-utility plans to be formed.

Some important factors were not included in the analysis. In particular, only single-step plans are evaluated, and only directly observable properties of the environment (i.e., sensory inputs) are learned. The result of these limitations is that PAGODA, using GDL as it currently stands, is only able to select features that allow relatively short-term plans to be formed. We present below some approaches to overcoming these limitations by expanding the analysis.

Additionally, the cost of adding new learning goals is not computed in the present system. It is clear from the tests in Chapter 7 that learning additional goals has a significant impact on the system's computational costs. Since the current implementation of PAGODA is not real-time (the learning algorithm is always allowed to run to completion), there is no negative effect on utility. However, in actual resource-bounded systems, this negative effect will have to be determined. Reserch on decision-theoretic approaches to real-time control such as metareasoning (Section 9.4) may provide some insights into the problem of determining and controlling the costs of learning.

Value of Information The computation of the value of knowing a feature only looks at the immediate effect of predicting the feature on the agent's utility. Specifically, it gives the expected *immediate* utility of knowing the feature. The value-of-information computation should be extended to consider explicitly the expected *future* utility of learning the feature, by considering intermediate effects caused by the increased ability to predict other learning goals that may then lead to increased utility.

For example, if the agent had learned how to predict when it would be at food (namely: when it was near food and moved towards the food), predicting when it would be near food would allow it to plan further ahead, thus maximizing longer-term utility. Under the single-step assumption, predicting near-food has no value because the single-step plans that can be formed given near-food do not have high utility: a two-step plan is needed to achieve the utility increase from eating food.

One way of achieving this would be to propagate the utility of learning a

feature backwards. Suppose a previously formed learning goal F (at-food in the above example) had value $V(F)$, representing the utility gain from knowing F. A new learning goal G (near-food above) contributes to utility in two ways: first, it may allow better immediate predictions about utility, leading to improved single-step plans. This contribution is represented by $V(G)$. Second, if G can be used to predict F, the agent can plan to achieve the value of G that will then allow it to select the value for F that maximizes a two-step plan. The effect of knowing G on predicting F can be measured by the decrease in variability of F when G is known. The decrease in variability (or, equivalently, increase in predictability) is given by the uniformity of F given G, minus the prior uniformity of F ($U(F|G) - U(F)$). The two-step plan can be improved by this difference times the utility that would be gained if F were predicted precisely.

Generating Internal States The agent should be able to hypothesize and learn about unobservable properties of the environment that affect its sensory inputs indirectly, by using its memory to preserve state. For example, PR has no direct way to determine that it is standing on food (it can smell the food, but can't distinguish its smell from other nearby food). If it creates a new term (which we refer to as an "internal state"), `at-food`(t), that it maintains by turning it on whenever it sees food and moves forward, and turns off after eating or moving off of the food, he can form better plans. Of course, it will have to learn when to turn it on and off, and decide when such an internal state may be useful at all.

Internal states may be useful when an action has different effects in what appears to be identical states. If the agent cannot find a description of its sensory inputs to distinguish between the two outcomes, there may be some hidden feature of the environment causing the different outcomes. (On the other hand, it could simply be some random effect, or noise in the inputs or motor actions of the agent, in which case no good theory will be found to predict the hypothesized internal states.) By forming an internal state (a new feature) that is true when one outcome occurs at time $t+1$, and false when the other outcome occurs, the agent can retroactively check what happened at the previous time step ($t-1$) and use inductive techniques to form

a theory to predict the intervening internal state.

We have developed an approach that would allow PR to generate internal states automatically in deterministic domains; the method is described in Appendix C. For probabilistic domains, uniformities will have to be used rather than determinations, and probabilistic analysis will be necessary to determine when and whether to add the internal states. To fully incorporate the technique into PAGODA, the agent will have to be able to determine the value of knowing such an internal state; the bias-evaluation method will have to be modified to allow internal states on either side of a rule; and the learning module will have to be able to learn theories about the internal states.

9.2 Selecting a Learning Bias

The bias evaluation technique we have presented is useful in domains for which uniformities, a learning curve, and the time-preference function are known. To make the technique more widely applicable, uniformities can be learned by the agent, the analysis can be extended to cover other types of background knowledge and biases, and learning curves and time-preference functions can be found for a variety of domains and prediction tasks.

If PAGODA's theories do not make good predictions, its representation may be insufficient to form a good model of the world. The agent should be able to recognize this situation, and shift bias when appropriate.

Finally, the thesis does not discuss the problem of searching the space of biases to find candidate biases. In the case of a large feature space, this problem will have to be addressed.

Background Knowledge Uniformities are simply a form of probabilistic knowledge to which inference methods and learning techniques can be applied. Automated methods for reasoning with uniformities (e.g., chaining ($U(X|Y)$ and $U(Y|Z)$ yield $U(X|Z)$) and combining ($U(X|Y)$ and $U(X|Z)$ yield $U(X|Y \wedge Z)$)) would allow the system to determine values for biases that aren't explicitly represented as uniformities.

Additionally, the probabilistic learning techniques of Chapter 6 should be extended to learn uniformities. Being able to learn uniformities would help to make the system more effective as an autonomous system, since it could learn background knowledge to use for later learning tasks, thus generalizing its previous learning experience.

In addition to uniformities, other types of background knowledge may be available or learnable, such as knowledge about relevance or independence, qualitative theories, and partial domain theories. The value-of-bias analysis should be extended to compute the estimated accuracy of various biases, given a variety of forms of background knowledge.

Another general type of background knowledge includes knowledge about operationality (i.e., cost of evaluation of features in the domain). Including operationality information would require extending the value-of-bias computation to compute expected *utility* of learned theories using a given bias, rather than simply their expected accuracy.

Types of Bias The analysis given in Chapter 5 focused only on selecting feature sets, and not on other aspects of the learning bias such as syntactic structure or feature value hierarchies.

Evaluating syntactic bias may require additional knowledge or assumptions to be evaluated in our framework. For example, suppose the agent wishes to determine whether (for a given feature set) to use k-DNF or k+1-DNF. The speed of convergence can be computed since we know the V-C dimension, but uniformities do not give the relative expected accuracy. Either some other form of background knowledge that specifies these expected accuracies, or a general method for estimating expected accuracy of various syntactic biases given a uniformity, is needed; once the expected accuracy is found, it can be incorporated directly into the analysis given in the thesis.

Tree-structured bias [Russell, 1988] may be useful in many domains to constrain the syntactic structure of theories. Russell's paper discusses tree-structured bias resulting from combining determinations. Since we are using uniformities, we need methods for combining them, and for evaluating and representing the resulting

tree-structured bias.

Another useful type of bias is internal disjunctions, represented as feature value hierarchies for categorical variables or ranges for numerical variables. Determining these in advance can help significantly in guiding learning. One question of particular interest is: if the agent has a particular object stored in multiple hierarchies, which should be used in a particular learning task? This problem is equivalent to determining the relevance of generalizations to a concept to be learned.

For example, the values of the property "color" may be organized into one hierarchy that generalizes hue (red, blue, green) and another that generalizes intensity (bright, dark, drab). Learning whether or not an object is edible may entail using the former hierarchy; learning whether or not an object is manufactured may require the latter. Some learning tasks may require both; for others, neither may be appropriate.

If one of the stored hierarchies has been more useful in the past for a particular sort of learning task, the agent should use it again, just as it reuses features that have proved relevant in the past. The agent will need an appropriate description of the hierarchies and a representation for background knowledge about the relevance of hierarchies, and perhaps knowledge about the hierarchies themselves (e.g., characterizations of learning tasks for which they have been useful, which can be generalized to predict future areas of relevance).

Value of Bias We have relied on particular estimates of the learning curve $q(t)$ and time-preference function $\mathcal{T}(t)$ for computing the value of a bias. A precise characterization of domains would allow the time-preference function to be chosen according to the properties of the domain. Similarly, characterization of learning algorithms (e.g., some have better techniques for eliminating irrelevant attributes; some require certain restricted syntactic biases) would allow $q(t)$ to be determined automatically. Alternatively, empirical tests may be run using a particular learning algorithm to determine its actual learning curve.

In any case, the agent should be able to dynamically modify both the time-preference function (for example, if its long-term behavior is poor, it may wish to increase the discounting parameter γ) and the learning curve (by examining its actual

learning behavior).

Shift of Bias PBE provides a method for selecting learning biases before learning has started. A completely autonomous agent will need to shift bias when it decides that an alternative bias is expected to perform better.

One approach to shifting bias would be to examine the current theory and try to determine why a predictive failure occurred. For example, in a UPT such as those learned by PAGODA, if all of the rules make poor predictions, it may be a syntactic restriction of the bias that is problematic (e.g., allowing disjunctions may be necessary). On the other hand, if one particular rule is failing to distinguish what appear to be significantly different situations, adding a new feature to the domain may be more appropriate (e.g., an internal state as defined in Section 9.1). The current theory and unexplained data can be used to guide the search for an appropriate internal state.

In a deterministic world, an agent expects to be able to learn complete and correct concepts. When the current concept space does not contain a theory which is consistent with *all* past observations, the agent's only option (other than failure) is to expand its concept space by relaxing either its syntactic bias (e.g., allowing disjunctions) or its semantic bias (e.g., including features that were previously thought to be irrelevant).

In the nondeterministic case, the agent does not expect to find a consistent theory. Instead, the signal that a different representation *might* be preferred is given by a failed expectation—i.e., any time the most likely outcome does not occur. In this case, the agent may settle for the imperfect theory, try to find a better theory under the current bias, or choose an alternative bias. Which of these options to choose will depend on several factors, including how accurate the best theory in the current language is, what the alternative representations are and the agent's past experience with representation shift.

In PBE, if the current bias is not performing as well as expected, it should be re-evaluated and compared to alternative biases. There is still a tradeoff involved: although the accuracy is lower than expected, the learning effort has already been

expended, so the expected learning curve is flatter than for a new bias.

The current bias is expected to reach an accuracy of $\hat{p}$ after $\hat{m}$ instances, as given in Equations 5.7 and 5.8. In order to avoid shifting bias too frequently, the agent must wait until the learning process appears to have actually converged (i.e., to have stabilized on a single best theory) or until a large number of instances have been processed. The value of the current bias can then be computed, using the actual accuracy and a flat learning curve (equivalent to assuming that the theory will get no better), and compared to the value of the next best bias. If the alternative bias' value is higher, representation shift should occur, and learning should continue with the new bias.

An alternative method would be to make bias shift an intrinsic part of the learning process: initially learn with the best bias, or simply with the single most relevant feature. Each time learning converges using one bias, the next most relevant feature would be added and learning restarted with the new bias. The old theory could be used for predictions until the accuracy of the new theory appeared to have surpassed the old one.

A number of variations on this approach are possible: if the agent has enough computational resources available, it can learn with several biases simultaneously and use whichever theory appears the best. This approach would provide the "envelope" of the learning curves of all of the biases, but would require significantly more processing time than using a single bias. Another approach would be to use the theory learned under the previous bias to "seed" the search in the new bias. Any savings, or *transference*, that can be gained by doing this will speed the search process and improve overall accuracy.

The advantage of this general approach is that initial learning quickly converges to reasonably good accuracy, and later learning allows the learner eventually to reach an optimal prediction level. The disadvantage is that if no transference occurs between biases, the cost of learning can become very high. Additionally, the questions of which biases to use at each stage, and how to determine when convergence has occurred, still must be answered.

Searching the Bias Space We have not addressed the problem of searching what may potentially be a very large space of biases: we have assumed that we can simply evaluate all alternative biases and choose the best. However, this will not be feasible in complex domains. Therefore, heuristics for searching the bias space are necessary.

First, operators for generating "neighboring" biases must be defined. Doing this requires defining the bias space and using the observed data to guide the search process (as discussed in the previous section). Second, an evaluation function must be defined: this could be the bias value itself, or an approximation, if the bias-value computation is too expensive.

9.3 Probabilistic Learning

Chapter 6 describes PAGODA's formalism for representing and learning probabilistic knowledge using a language that combines the advantages of first-order logic and probability. The utility of the formalism lies in the inference method; however, applying the inference method as it currently stands requires the theory to be of a restricted form. Additionally, a number of problems relating to the learning process have not been fully addressed.

Constraints on Theories The constraints on theories are overly restrictive in some ways: although they allow certain kinds of independence to be captured automatically, it may be desirable to allow more complex interactions. In general, the problem of deriving a complete conditional probability distribution, given partial knowledge of the conditional probabilities, is non-trivial. Our model is more complete than, for example, the noisy-or model used by [Pearl, 1986] and others, but still not able to represent all possible interactions. Maximum entropy techniques provide a theoretically sound method for filling in all of the conditional probabilities, given any subset, but are intractable in the general case. Identifying common types of interactions and providing general solutions for computing the effects of those interactions is necessary.

Evaluating Theories Using point probabilities (i.e., a single numerical value) to evaluate theories causes the agent to lose useful information. Second-order probabilities, giving certainty values on the first-order point probabilities, or probability intervals provide additional information that would allow the agent to determine how likely it is that the current best theory is really better than the alternatives. This knowledge can be used to guide experimentation and exploration in the planning process: if the agent believes it to be likely that some alternative theory is better than the current best theory, it should attempt to collect data to decrease the uncertainty.

Another factor that should be considered when evaluating and selecting theories is the cost of errors. For example, if a false positive (predicting that a property holds when it does not) has higher cost than a false negative, then theories that are less likely to make false positives should be preferred. Additionally, the cost of using the theories should be taken into account. For example, if some information is expensive to gather and only reduces uncertainty slightly, it may not be worthwhile. Decision theory can be used to select the theory with highest overall utility (rather than simply selecting the most probable theory, as PAGODA currently does).

Searching the Space of Theories PAGODA's current search techniques for splitting and merging rules are essentially *ad hoc*. The theories generated depend on the order in which observations arrive, the number of candidate theories maintained, and the simplicity metric used for evaluation. In some cases, the best theory may never be found.

Heuristic search for theories is a difficult problem, complicated by the fact that every new observation changes the probability of theories, which changes the evaluation function over the theory space. Most existing search techniques assume a static space to be searched, in which the evaluation function does not change. The convergence analysis of search techniques in dynamic domains changes drastically, and has not been addressed in depth by either heuristic search or machine learning researchers.

Incremental Learning Incremental learning techniques process new observations (updating the current theory) as they arrive, in contrast to batch learning algorithms which find a theory to describe a set of previously collected observations. We use a stronger definition of incremental learning techniques, which requires that the learning method much be resource bounded in both space and time. In order to satisfy this condition, an incremental learning method cannot simply store every observation and re-run an essentially batch algorithm as each observation arrives.

The heuristic search method PAGODA currently uses is not incremental by this definition, as it stores all of the observations (requiring a potentially unbounded amount of memory) and reprocesses them when the theories are modified (requiring unbounded computation time).

Most other existing "incremental learning techniques" also fail to satisfy our definition of incremental learning. Although they generate a new theory after each new observation, the updating technique often requires examining all of the previous instances, and almost always requires storing all of the instances. For example, Utgoff's ID5 [1988], an incremental decision-tree-building algorithm, stores all of the observed instances at the appropriate leaf and uses them to decide whether to split or merge subtrees. COBWEB, a conceptual clustering method [Fisher, 1987b], uses a similar technique.

We believe that these "quasi-incremental" approaches will be too expensive for agents with limited resources which must operate in complex domains. However, many problems arise when designing truly incremental learning algorithms. In particular, if previous instances are not all stored, the agent cannot know with certainty how many instances would have been assigned to a newly-formed rule. Heuristics for estimating the probabilities of these new rules must be developed and analyzed theoretically and empirically. For example, one heuristic might be to assign a percentage of instances proportional to the size of the subspaces formed when splitting a rule, or to store only a limited number of "boundary examples" representing a set of particularly important examples [Haussler, 1988]. The adequacy of these approaches will depend on the representation, inference mechanism, and search techniques used for learning.

9.4 Probabilistic Planning

PAGODA currently uses a fairly simple forward-chaining mechanism to compute expected utilities of sequences of actions, and selects either the action that leads to highest expected utility or a random action with a fixed probability (representing the level of exploration).

Probabilistic planning (i.e., planning using a probabilistic world model) is a relatively unexplored field. "Classical" planning requires a deterministic world model. Reactive planning, a more recent approach, uses hand-tuned production rules rather than performing deliberative planning from a world model.

The open questions in this field include:

- How should the utility of future rewards be weighted?
- How can degrees of belief in learned theories be incorporated into the planning process?
- How should the agent balance apparently-optimal behavior (i.e., maximizing expected utility according to the current world model) with experimentation and exploration to refine and correct the model?
- How should the agent allocate its time between planning, learning, and acting?

Discounting, weighting predictions from multiple theories, computing a value of curiosity, and metareasoning are methods that may be useful in solving these problems.

Discounting A time-preference model such as that used in PAGODA's probabilistic bias evaluation technique, expressing the degree to which the agent is willing to wait for long-term rewards, could be used to compute an expected *discounted* future utility of action sequences. The time-preference model should take into account factors such as uncertainty of predictions (which becomes exponentially greater as more predictions are chained together), changeability of the environment, and life expectancy. Discounting must be integrated into the planning process, and used with metareasoning (see below) to control search time.

Weighting Predictions According to Bayesian probability theory, rather than simply using the predictions of the best theory (as PAGODA currently does), the predictions of all potential theories should be combined, weighted by their probabilities, to get a correct expectation. However, using all theories is computationally infeasible. The agent should combine the predictions of several of the current best theories. A formal analysis to determine the probability of correctness, given the probability of the theories and possibly second-order probabilities reflecting the degree of belief in the first-order probabilities, can be done using techniques from computational learning theory.

Value of Curiosity The planner described in the thesis takes a random action with fixed probability, determined by the user. A preferable approach would be to develop a theoretically justifiable technique for deciding whether to explore. For example, second-order probabilities on current theories could be used to determine the rate of exploration: the higher the uncertainty, the more exploration should be done. Alternatively, rather than selecting either the best action or a random actions, actions could be selected with probability determined by their expected utility (using, for example, a a Boltzmann distribution). This problem is essentially equivalent to the n-armed bandit problem discussed in Section 2.3.4.

Metareasoning In real-time systems, an agent must control the time spent computing expected utility. Choosing an "optimal" action is not useful if finding this optimal action takes so long that the agent is eaten before it is found.

Metareasoning—reasoning about the relative utility of computational and external actions—may prove useful in controlling the deliberative behavior of the agent. This involves more than just deciding whether to plan or act. For example, the metareasoner may control search so that only the most promising action sequences are explored, or it may decide to cache plan knowledge by compiling the learned world model into situation-action rules to be applied in the future [Russell, 1989].

9.5 Conclusions

We have provided a model of learning in autonomous domains that integrates solutions to the problems of deciding what to learn, selecting learning biases, representing and learning probabilistic theories, and planning with learned probabilistic knowledge. The interactions among these problems have been considered throughout; because of this, our approach is more complete than previous models.

In particular, we have developed a representation (UPTs) and inference method (PBE) for probabilistic world models, a mechanism for autonomous agents to decide how to focus their attention in complex learning environments (GDL), an innovative technique for finding a learning bias in probabilistic domains (PBE), and a Bayesian evaluation technique for probabilistic theories.

Still, building general intelligent agents is an extremely difficult long-term goal; accordingly, we have discussed some of the most pressing open issues in intelligent agent design. PAGODA is the result of identifying and considering the issues involved in agent design as they relate to one another, and combining old and new technologies and ideas in a coherent agent model.

One of the most important open issues in machine learning is incremental, resource-bounded learning. How do we guide the search for good theories in complex, nondeterministic domains, when the only evidence we have of the true theory is a limited sample, and we cannot afford to remember and reprocess all of the observed instances?

The single most important issue that the machine learning community must address, though, is the use of knowledge to constrain learning. If we wish to build agents that can operate in real time in complex, uncertain environments, the agents must be able to use prior knowledge (gained from previous interactions with the environment and with other agents) to make future learning faster and more effective. Agents must not simply learn—they must learn to learn better.

Bibliography

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *AAAI*, pages 268–272, 1987.

[Anderson, 1980] John R. Anderson. *Cognitive Psychology and Its Implications.* W. H. Freeman and Company, 1980.

[Angluin and Laird, 1986] Dana Angluin and P. D. Laird. Identifying k-CNF formulas from noisy examples. Technical Report YALEU/DCS/TR-478, Yale University, June 1986.

[Angluin and Smith, 1983] Dana Angluin and Carl H. Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15(3):237–269, September 1983.

[Babcock *et al.*, 1990] Marla S. Babcock, Wilma K. Olson, and Edwin P. D. Pednault. The use of the Minimum Description Length principle to segment DNA into structural and functional domains. In *Working Notes: AAAI Spring Symposium on the Theory and Application of Minimal-Length Encoding*, pages 40–44, 1990.

[Bacchus *et al.*, 1988] Fahiem Bacchus, Henry Kyburg, Jr., and Mariam Thalos. Against conditionalization. Technical Report 256, University of Rochester Computer Science Dept., June 1988.

[Bacchus, 1987] Fahiem Bacchus. Statistically founded degrees of belief. Technical Report 87-102, University of Alberta, 1987.

[Bacchus, 1990] Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities.* MIT Press, 1990.

[Berry and Fristedt, 1985] Donald A. Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, 1985.

[Birnbaum and Collins, 1991] Lawrence A. Birnbaum and Gregg C. Collins, editors. *Proceedings of the Eighth International Workshop on Machine Learning.* Morgan Kaufmann, 1991.

[Blumer *et al.*, 1986] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proc. 18th ACM Symposium on Theory of Computation*, pages 273–282, 1986.

[Blumer *et al.*, 1987] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

[Board and Pitt, 1989] Raymond Board and Leonard Pitt. On the necessity of Occam algorithms. Technical Report UIUCDCS-R-89-1544, UIUC, September 1989.

[Brooks, 1982] Rodney A. Brooks. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, 1(4):29–68, 1982.

[Brooks, 1991] Rodney A. Brooks. Intelligence without reason. In *IJCAI*, 1991. Computers and Thought Lecture.

[Bundy *et al.*, 1985] Alan Bundy, Bernard Silver, and Dave Plummer. An analytical comparison of some rule-learning programs. *Artificial Intelligence*, 27, 1985.

[Buntine, 1990] Wray Buntine. *A Theory of Learning Classification Rules.* PhD thesis, University of Technology, Sydney, February 1990.

[Carbonell and Gil, 1987] Carbonell and Gil. Learning by experimentation, 1987.

[Carbonell and Hood, 1986] Jaime Carbonell and Greg Hood. The World Modelers Project: Objectives and simulator architecture. In Tom Mitchell, Jaime Carbonell, and Ryszard Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 29–34. Kluwer Academic Publishers, 1986.

[Carbonell *et al.*, 1983] Jaime Carbonell, Ryszard Michalski, and Tom Mitchell. An overview of machine learning. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning.* Morgan Kaufman, 1983.

[Carnap, 1950] Rudolf Carnap. *Logical Foundations of Probability.* University of Chicago Press, 1950.

[Chaitin, 1975] Gregory J. Chaitin. A theory of program size formally identical to information theory. *JACM*, 22(3):329–340, July 1975.

[Chaitin, 1977] G. J. Chaitin. Algorithmic information theory. *IBM J. Res. Develop.*, 21:350–359, July 1977.

[Chapman and Kaelbling, 1990] David Chapman and Leslie Pack Kaelbling. Learning from delayed reinforcement in a complex domain. Technical Report TR-90-11, Teleos Research, December 1990.

[Cheeseman, 1988] Peter Cheeseman. An inquiry into computer understanding. *Computational Intelligence*, 4(1):58–66, 1988.

[Cooper and Herskovits, 1991] Gregory F. Cooper and Edward Herskovits. A Bayesian method for constructing Bayesian belief networks from databases. In *Workshop on Uncertainty in Artificial Intelligence*, pages 86–94, 1991.

[Cover, 1985] Thomas M. Cover. Kolmogorov complexity, data compression, and inference. In J. K. Skwirzynski, editor, *The Impact of Processing Techniques on Communications*, pages 23–33. Martin Nijhoff, 1985.

[Cox, 1946] R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14:1–13, 1946.

[Davies and Russell, 1987] Todd Davies and Stuart Russell. A logical approach to reasoning by analogy. Technical Report Note 385, AI Center, SRI International, July 1987.

[Doyle, 1988] Jon Doyle. On rationality and learning. Technical Report CMU-CS-88-122, CMU, March 1988.

[Doyle, 1990] Jon Doyle. Rationality and its roles in reasoning. In *AAAI*, pages 1093–1100, 1990.

[Drummond and Bresina, 1990] Mark Drummond and John Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *AAAI*, pages 138–144, 1990.

[Fisher and Langley, 1985] Douglas Fisher and Pat Langley. Approaches to conceptual clustering. In *IJCAI*, pages 691–697, 1985.

[Fisher, 1987a] Douglas Fisher. Improving inference through conceptual clustering. In *AAAI*, pages 461–465, 1987.

[Fisher, 1987b] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning Journal*, 2:139–172, 1987.

[Fu and Buchanan, 1985] Li-Min Fu and Bruce G. Buchanan. Learning intermediate concepts in constructing a hierarchical knowledge base. In *IJCAI*, pages 659–666, 1985.

[Fung and Crawford, 1990] Robert M. Fung and Stuart L. Crawford. Constructor: A system for the induction of probabilistic models. In *AAAI*, pages 762–769, 1990.

[Getoor, 1989] Lise Getoor. The instance description: How it can be derived and the use of its derivation, 1989. MS thesis, UC Berkeley.

[Gil, 1991] Yolanda Gil. A domain-independent framework for effective experimentation in planning. In *Machine Learning Workshop*, pages 13–17, 1991.

[Gold, 1967] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Goldman and Charniak, 1990] Robert P. Goldman and Eugene Charniak. Dynamic construction of belief networks. In *Proc. of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97, 1990.

[Good, 1983] I. J. Good. *Good Thinking.* University of Minnesota Press, 1983.

[Goodman and Smyth, 1989] Rodney Goodman and Padhraic Smyth. The induction of probabilistic rule sets: The ITRULE algorithm. In *Machine Learning Workshop*, pages 129–132, 1989.

[Goodman, 1955] Nelson Goodman. *Fact, Fiction, and Forecast.* Harvard University Press, 1955.

[Goodman, 1958] Nelson Goodman. The test of simplicity. *Science*, 128:1064–1069, 1958.

[Goodman, 1983] Nelson Goodman. *Fact, Fiction, and Forecast (4/e).* Harvard University Press, 1983.

[Grosof and Russell, 1989] Benjamin N. Grosof and Stuart J. Russell. Shift of bias as non-monotonic reasoning. Technical Report RC 14620 (No. 64608), IBM Research Division, February 1989.

[Hacking, 1975] Ian Hacking. *The Emergence of Probability.* Cambridge University Press, 1975.

[Halpern, 1989a] Joseph Y. Halpern. An analysis of first-order logics of probability (revised version). Technical Report RJ 6882, IBM Almaden Research Center, June 1989.

[Halpern, 1989b] Joseph Y. Halpern. Knowledge, probability, and adversaries. Technical Report RJ 7045, IBM Almaden Research Center, September 1989.

[Harper *et al.*, 1981] William L. Harper, Robert Stalnaker, and Glenn Pearce (eds.). *IFS: Conditionals, Belief, Decision, Chance, and Time.* D. Reidel, 1981.

[Haussler *et al.*, 1990] D. Haussler, N. Littlestone, and M. Warmuth. Predicting {0,1}-functions on randomly drawn points. Technical Report UCSC-CRL-90-54, U.C. Santa Cruz Computer Science Laboratory, December 1990.

[Haussler *et al.*, 1991] D. Haussler, M. Kearns, and R. E. Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension, 1991. Preliminary draft for distribution at *Neural Networks for Computing.*

[Haussler, 1987] David Haussler. Bias, version spaces and Valiant's learning framework. In *Machine Learning Workshop*, pages 324–336, 1987.

[Haussler, 1988] David Haussler. Space efficient learning algorithms. Technical Report UCSC-CRL-88-2, UC Santa Cruz, March 1988.

[Haussler, 1989] David Haussler. Generalizing the PAC model for neural net and other learning applications. Technical Report UCSC-CRL-89-30, UC Santa Cruz, September 1989.

[Hempel, 1952] C. G. Hempel. *Fundamentals of Concept Formation in Empirical Science.* University of Chicago Press, 1952.

[Holland *et al.*, 1986] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction: Processes of Inference, Learning, and Discovery.* MIT Press, 1986.

[Holtzman, 1989] Samuel Holtzman. *Intelligent Decision Systems.* Addison-Wesley, 1989.

[Horsch and Poole, 1990] Michael C. Horsch and David Poole. A dynamic approach to probabilistic inference using Bayesian networks. In *Proc. of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 155–161, 1990.

[Horvitz and Heckerman, 1986] Eric Horvitz and David Heckerman. The inconsistent use of measures of certainty in artificial intelligence research. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 137–151. North-Holland, 1986.

[Horvitz *et al.*, 1989] Eric J. Horvitz, Gregory F. Cooper, and David E. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *IJCAI*, pages 1121–1127, 1989.

[Hume, 1975] David Hume. *Enquiries Concerning Human Understanding.* Clarendon Press, 1975.

[Hunter, 1986] Daniel Hunter. Uncertain reasoning using maximum entropy inference. In *Uncertainty in Artificial Intelligence*, pages 203–209. Elsevier, 1986.

[Kaelbling, 1990] Leslie Pack Kaelbling. *Learning in Embedded Systems.* PhD thesis, Stanford University, 1990.

[Kanazawa and Dean, 1989] Keiji Kanazawa and Thomas Dean. A model for projection and action. In *IJCAI*, pages 985–990, 1989.

[Kearns and Li, 1987] Michael Kearns and Ming Li. Learning in the presence of malicious errors (preliminary report). Technical Report TR-03-87, Harvard University, 1987.

[Kedar-Cabelli, 1986] Smadar Kedar-Cabelli. Purpose-directed analogy: A summary of current research. In Tom Mitchell, Jaime Carbonell, and Ryszard Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 123–126. Kluwer Academic Publishers, 1986.

[Kedar-Cabelli, 1987] Smadar Kedar-Cabelli. Formulating concepts according to purpose. In *AAAI*, pages 477–481, 1987.

[Kemeny, 1963] John G. Kemeny. Carnap on probability of induction. In *The Philosophy of Rudolf Carnap (The Library of Living Philosophers, v. 11)*, pages 711–738. Open Court, 1963.

[Korf, 1980] Richard Korf. Toward a model of representation changes. *Artificial Intelligence*, 14:41–78, 1980.

[Kuhn, 1962] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, 1962.

[Kuipers, 1985] Benjamin J. Kuipers. The Map-Learning Critter. Technical Report AITR85-17, University of Texas, Austin, December 1985.

[Kyburg, 1974] Henry E. Kyburg. *The Logical Foundations of Statistical Inference.* Reidel, 1974.

[Laird and Rosenbloom, 1990] John E. Laird and Paul S. Rosenbloom. Integrating execution, planning, and learning in soar for external environments. In *AAAI*, pages 1022–1029, 1990.

[Langley *et al.*, 1983] Pat Langley, Gary L. Bradshaw, and Herbert A. Simon. Rediscovering chemistry with the BACON system. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning*, pages 307–329. Morgan Kaufman, 1983.

[Langley *et al.*, 1986] Pat Langley, Jan M. Zytkow, Herbert A. Simon, and Gary L. Bradshaw. The search for regularity. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning II*, pages 425–470. Morgan Kaufman, 1986.

[Langley *et al.*, 1987] Pat Langley, Herbert A. Simon, Gary L. Bradshaw, and Jan M. Zytkow. *Scientific Discovery: Computational Explorations of the Creative Process.* MIT Press, 1987.

[Lebowitz, 1986a] Michael Lebowitz. Concept learning in a rich input domain: Generalization-based memory. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning II*, pages 193–214. Morgan Kaufman, 1986.

[Lebowitz, 1986b] Michael Lebowitz. Integrated learning: Controlling explanation. *Cognitive Science*, 10(2):219–240, 1986.

[Lebowitz, 1986c] Michael Lebowitz. Not the path to perdition: The utility of similarity-based learning. In *AAAI*, pages 533–537, 1986.

[Lenat and Brown, 1984] Douglas B. Lenat and John Seely Brown. Why AM and EURISKO appear to work. *Artificial Intelligence*, 23:269–294, 1984.

[Lenat, 1979] D. B. Lenat. On automated scientific theory formation: A case study using the AM program. In J. E. Hayes, D. Michie, and L. I. Mikulich, editors, *Machine Intelligence 9*, pages 251–283. Horwood, 1979.

[Lenat, 1982a] Douglas B. Lenat. Eurisko: A program that learns new heuristics and domain concepts. Technical Report HPP-82-26, Stanford University, 1982.

[Lenat, 1982b] Douglas B. Lenat. Theory formation by heuristic search. Technical Report HPP-82-25, Stanford University, 1982.

[Levine and Tribus, 1979] Raphael D. Levine and Myron Tribus. *The Maximum Entropy Formalism Conference.* MIT Press, 1979.

[Li and Vitányi, 1989] Ming Li and Paul M. B. Vitányi. Inductive reasoning and Kolmogorov complexity. In *IEEE Structure in Complexity Theory Conference*, 1989.

[Lozano-Pérez *et al.*, 1984] T. Lozano-Pérez, M. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.

[Martin and Billman, 1991] Joel D. Martin and Dorrit O. Billman. Variability bias and category learning. In *Machine Learning Workshop*, pages 90–94, 1991.

[Medin *et al.*, 1987] Douglas L. Medin, William D. Wattenmaker, and Ryszard S. Michalski. Constraints and preferences in inductive learning: An experimental study of human and machine performance. *Cognitive Science*, 11:299–339, 1987.

[Michalski and Stepp, 1983] Ryszard S. Michalski and Robert E. Stepp. Learning from observation: Conceptual clustering. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning*, pages 331–364. Morgan Kaufman, 1983.

[Michalski, 1980] Ryszard S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, 4(3):219–244, 1980.

[Mitchell and Keller, 1983] T. Mitchell and R. Keller. Goal directed learning. In *Second International Machine Learning Workshop*, pages 117–118, 1983.

[Mitchell *et al.*, 1989] Tom M. Mitchell, Matthew T. Mason, and Alan D. Christiansen. Toward a learning robot. Technical Report CMU-CS-89-106, CMU, January 1989.

[Mitchell, 1980] Tom Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, May 1980.

[Muggleton and Buntine, 1988] Stephen Muggleton and Wray Buntine. Machine invention of first-order predicates by inverting resolution. In *Machine Learning Conference*, pages 339–352, 1988.

[Muggleton, 1987] Stephen Muggleton. Duce, an oracle based approach to constructive induction. In *IJCAI*, pages 287–292, 1987.

[Muggleton, 1988] Stephen Muggleton. A strategy for constructing new predicates in first order logic. In *EWSL*, pages 123–130, 1988.

[Neufeld and Poole, 1988] Eric Neufeld and David Poole. Combining logic and probability. *Computational Intelligence*, 4(1):98–99, 1988.

[Nilsson, 1986] Nils J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[Parr *et al.*, 1992] Ronald Parr, Stuart Russell, and Mike Malone. The RALPH system. Technical report, UC Berkeley, 1992. (Forthcoming).

[Pazzani *et al.*, 1987] Michael Pazzani, Michael Dyer, and Margot Flowers. Using prior learning to facilitate the learning of new causal theories. In *IJCAI*, pages 277–279, 1987.

[Pearl, 1978] Judea Pearl. On the connection between the complexity and credibility of inferred models. *Int. J. General Systems*, 4:255–264, 1978.

[Pearl, 1986] Judea Pearl. A constraint-propagation approach to probabilistic reasoning. In *Uncertainty in Artificial Intelligence*, pages 357–369. Elsevier, 1986.

[Pearl, 1988a] Judea Pearl. On logic and probability. *Computational Intelligence*, 4(1):99–103, 1988.

[Pearl, 1988b] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[Pednault, 1988] Edwin P. D. Pednault. Inferring probabilistic theories from data. In *AAAI*, pages 624–628, 1988.

[Pednault, 1989] Edwin P. D. Pednault. Some experiments in applying inductive inference principles to surface reconstruction. In *IJCAI*, pages 1603–1609, 1989.

[Pollard, 1969] Arnold Bruce Pollard. *A Normative Model for Joint Time/Risk Preference Decision Problems.* PhD thesis, Stanford Engineering–Economic Systems Department, 1969.

[Qi and Poole, 1991] Runping Qi and David Poole. High level path planning with uncertainty. In *Workshop on Uncertainty in Artificial Intelligence*, pages 287–294, 1991.

[Quinlan, 1983] R. Quinlan. Learning efficient classification procedures and their application to chess end games. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning*, pages 463–482. Morgan Kaufman, 1983.

[Quinlan, 1986] R. Quinlan. The effect of noise on concept learning. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning II*, pages 149–166. Morgan Kaufman, 1986.

[Rendell *et al.*, 1987] Larry Rendell, Raj Seshu, and David Tcheng. Layered concept learning and dynamically-variable bias management. In *Machine Learning Conference*, pages 308–314, 1987.

[Rendell, 1985] Larry Rendell. Genetic plans and the Probabilistic Learning System: Synthesis and results. Technical Report UIUCDCS-R-85-1217, University of Illinois at Urbana-Champaign, 1985.

[Rendell, 1986] Larry Rendell. Induction, of and by probability. Technical Report UIUCDCS-R-86-1293, University of Illinois at Urbana-Champaign, 1986.

[Riddle, 1986] Patricia J. Riddle. Exploring shifts of representation. In Tom Mitchell, Jaime Carbonell, and Ryszard Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 275–280. Kluwer Academic Publishers, 1986.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Rissanen, 1983] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[Rissanen, 1986] Jorma Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986.

[Rissanen, 1987] Jorma Rissanen. Stochastic complexity. *J. R. Stat. Soc. B*, 49(3):223–239 and 252–265, 1987. With commentary.

[Ritchie and Hanna, 1984] G. D. Ritchie and F. K. Hanna. AM: A case study in A.I. methodology. *Artificial Intelligence*, 23(3):249–268, 1984.

[Rivest and Schapire, 1987] Ronald L. Rivest and Robert E. Schapire. A new approach to unsupervised learning in deterministic environments. In *Machine Learning Workshop*, pages 364–375, 1987.

[Rivest and Sloan, 1988] Ronald L. Rivest and Robert Sloan. A new model for inductive inference. In Moshe Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 13–27. Morgan Kaufmann, 1988.

[Rosch and Lloyd, 1978] Eleanor Rosch and B. B. Lloyd, editors. *Cognition and Categorization.* Erlbaum, 1978.

[Rosch *et al.*, 1976] Eleanor Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.

[Russell and Grosof, 1987] Stuart J. Russell and Benjamin N. Grosof. A declarative approach to bias in concept learning. In *AAAI*, pages 505–510, 1987.

[Russell and Wefald, 1991] Stuart Russell and Eric Wefald. *Do the Right Thing: Studies in Limited Rationality.* MIT Press, 1991.

[Russell, 1986] Stuart Jonathan Russell. *Analogical and Inductive Reasoning.* PhD thesis, Stanford University, 1986.

[Russell, 1988] Stuart J. Russell. Tree-structured bias. In *AAAI*, pages 641–645, 1988.

[Russell, 1989] Stuart J. Russell. Execution architectures and compilation. In *IJCAI*, pages 15–20, 1989.

[Savage, 1977] L. J. Savage. *The Foundations of Statistics.* Dover, 1977. 2nd rev. ed.

[Schlimmer and Granger, 1986] Jeffrey C. Schlimmer and Richard H. Granger, Jr. Beyond incremental processing: Tracking concept drift. In *AAAI*, pages 502–507, 1986.

[Schlimmer, 1987a] Jeffrey C. Schlimmer. Incremental adjustments of representation for learning. In *Machine Learning Workshop*, pages 79–90, 1987.

[Schlimmer, 1987b] Jeffrey C. Schlimmer. Learning and representation change. In *AAAI*, pages 511–515, 1987.

[Segen, 1986] Jakub Segen. Learning from data with errors. In Tom Mitchell, Jaime Carbonell, and Ryszard Michalski, editors, *Machine Learning: A Guide to Current Research*, pages 299–302. Kluwer Academic Publishers, 1986.

[Shachter *et al.*, 1990] Ross D. Shachter, Brendan A. Del Favero, and Bruce D'Ambrosio. Symbolic probabilistic inference in belief networks. In *AAAI*, pages 126–131, 1990.

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Shortliffe, 1976] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* American Elsevier, 1976.

[Smith, 1987] David E. Smith. A decision-theoretic approach to the control of planning search. Technical Report LOGIC-87-11, Stanford University, January 1987.

[Solomonoff, 1964a] R. J. Solomonoff. A formal theory of inductive inference, Part I. *Information and Control*, 7:1–22, 1964.

[Solomonoff, 1964b] R. J. Solomonoff. A formal theory of inductive inference, Part II. *Information and Control*, 7:224–254, 1964.

[Solomonoff, 1975] R. J. Solomonoff. Inductive inference theory: A unified approach to problems in pattern recognition and artificial intelligence. In *IJCAI*, pages 274–280, 1975.

[Solomonoff, 1986] Ray Solomonoff. The application of algorithmic probability to problems in artificial intelligence. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 473–491. North-Holland, 1986.

[Stepp and Michalski, 1986] Robert E. Stepp and Ryszard S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning II*, pages 471–498. Morgan Kaufman, 1986.

[Subramanian and Feigenbaum, 1986] Devika Subramanian and Joan Feigenbaum. Factorization in experiment generation. In *AAAI*, 1986.

[Subramanian and Genesereth, 1987] Devika Subramanian and Michael R. Genesereth. The relevance of irrelevance. In *IJCAI*, pages 416–422, 1987.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Conference*, 1990.

[Tan and Schlimmer, 1990] Ming Tan and Jeffrey C. Schlimmer. Two case studies in cost-sensitive concept acquisition. In *AAAI*, pages 854–860, 1990.

[Utgoff, 1986] Paul Utgoff. Shift of bias for inductive concept learning. In Ryszard Michalski, Jaime Carbonell, and Tom Mitchell, editors, *Machine Learning II*, pages 107–148. Morgan Kaufman, 1986.

[Utgoff, 1988] Paul E. Utgoff. ID5: An incremental ID3. In *Machine Learning Conference*, pages 107–120, 1988.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *CACM*, 27(11):1134–1142, November 1984.

[von Neumann and Morgenstern, 1947] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1947.

[Watanabe, 1960] S. Watanabe. Information-theoretical aspects of inductive and deductive inference. *IBM Journal of Research Development*, 4:208–231, 1960.

[Wise and Henrion, 1986] B. P. Wise and M. Henrion. A framework for comparing uncertainty inference systems to probability. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence.* North-Holland, 1986.

[Zadeh, 1980] L. A. Zadeh. Inference in fuzzy logic. In *IEEE Tenth Annual Symposium on Multiple-Valued Logic*, pages 124–131, June 1980.

Appendix A

Maximum Entropy Proof

We show here that the assumptions made in Section 5.4 to find the distribution of outcomes, given a uniformity, yield the maximum entropy distribution (using the uniformity as a constraint).

Recall that the assumptions were:

1. For each value of $\mathcal{F}$, there is one $\mathcal{O}$-value, $\hat{o}$, which occurs most often.

2. The other values of $\mathcal{O}$ occur equally often.

These two assumptions yield the probabilities

$$P(\mathcal{O}_1|\mathcal{F}) = \hat{p} \tag{A.1}$$

$$P(\mathcal{O}_i|\mathcal{F}) = \frac{1-\hat{p}}{n-1}, \quad i = 2, \ldots, n \tag{A.2}$$

A probability distribution $\mathbf{p}$ has entropy $H(\mathbf{p})$, where

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log p_i$$

The entropy is a measure of the randomness of the distribution; alternatively, it can be viewed as the amount of information contained in the distribution (a uniform distribution has less information than a skewed distribution).

The probability distribution that maximizes the entropy of the distribution, given a set of constraints, is the distribution that adds the least amount of informa-

tion to the constraints.[1] Maximum entropy is a standard technique for determining probabilities of events when a complete joint distribution is not available.

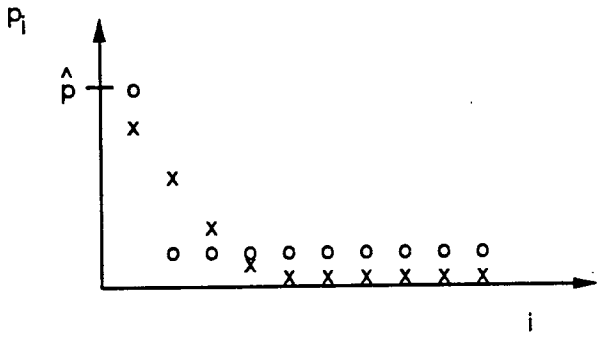

Figure A.1: Alternative Probability Distributions

Intuitively, it makes sense that the assumptions about the distribution of outcomes given above would maximize entropy, since they yield the flattest possible distribution. However, it does not seem implausible that a smooth curve ("x" in Figure A.1) would have higher entropy than the step curve ("o") yielded by the assumptions.

The constraints on the distribution are that the probabilities must sum to one, and that the sum of the squares of the probabilities is equal to the uniformity u. Mathematically, these constraints can be written as

$$\begin{aligned} g(\mathbf{p}) &= \sum_{i=1}^{n} p_i - 1 = 0 \\ h(\mathbf{p}) &= \sum_{i=a}^{n} p_i^2 - u = 0 \end{aligned}$$

Using the method of Lagrange multipliers to maximize entropy subject to the constraints g and h yields the Lagrange function

$$\begin{aligned} F(\mathbf{p}) &= H(\mathbf{p}) + \lambda_1 g(\mathbf{p}) + \lambda_2 h(\mathbf{p}) \\ &= -\sum p_i \log p_i + \lambda_1(\sum p_i - 1) + \lambda_2(\sum p_i^2 - u) \end{aligned}$$

[1]The information is minimized subject to a bias for maximally uniform distributions under the specified description language. In this case, maximum entropy will tend to favor distributions that are as flat as possible.

Any distribution that maximizes the entropy yields zero partial derivatives with respect to each p_i, λ_1, and λ_2. There are $n+2$ partial derivatives, as follows:

$$\begin{aligned} F_i &= \tfrac{\delta}{\delta p_i} F &&= \log p_i + 1 + \lambda_1 + \lambda_2(2p_i) \quad i = 1, \ldots, n \\ C_1 &= \tfrac{\delta}{\delta \lambda_1} F &&= \sum p_i - 1 \\ C_2 &= \tfrac{\delta}{\delta \lambda_2} F &&= \sum p_i^2 - u \end{aligned}$$

In principle, the maximum entropy distribution(s) can be found by solving these $n+2$ partial differential equations. However, since this is extremely difficult in practice, and all we wish to show is that our solution *does* maximize entropy, we simply demonstrate that the distribution given by the assumptions A.1 and A.2 yields a solution to the differential equations.

Equation 5.7 gives the value for $\hat{p}$ in terms of n and u:

$$\hat{p} = \frac{1 + \sqrt{1 - n + n(n-1)u}}{n}$$

Letting $k = \sqrt{1 - n + n(n-1)u}$ for convenience,

$$\begin{aligned} \hat{p} &= \frac{1+k}{n} \\ \hat{p}_i &= \frac{1-\hat{p}}{n-1} = \frac{n-(1+k)}{n(n-1)} \end{aligned}$$

Since the values for $\hat{p}_i$ were found in terms of $\hat{p}$ to satisfy constraint C_1, and the value for $\hat{p}$ was derived to satisfy constraint C_2 (the uniformity), we will not show that they are satisfied.

We are then left with the n constraints F_i. First we solve for λ_1 in terms of k and λ_2 using constraint F_1, then show that there is a value for λ_2 that satisfies the remaining equations (since they are all the same, and $p_2 \ldots p_n$ have identical values, this is straightforward).

Solving for λ_1 in F_1:

$$\begin{aligned} 0 &= \log \hat{p} + 1 + \lambda_1 + \lambda_2(2\hat{p}) \\ \lambda_1 &= -(1 + \log \hat{p} + 2\lambda_2 \hat{p}) \\ &= -\left(1 + \log\left(\frac{1+k}{n}\right) + 2\lambda_2\left(\frac{1+k}{n}\right)\right) \end{aligned}$$

Solving for λ_2 in F_i $(i = 2 \ldots n)$:

$$\begin{aligned} 0 &= \log p_i + 1 + \lambda_1 + \lambda_2(2p_i) \\ &= \log \frac{n-(1+k)}{n(n-1)} + 1 \\ &\quad - \left(1 + \log\left(\frac{1+k}{n}\right) + 2\lambda_2\left(\frac{1+k}{n}\right)\right) \\ &\quad + 2\lambda_2\left(\frac{n-(1+k)}{n(n-1)}\right) \end{aligned}$$

$$\begin{aligned} 2\lambda_2\left(\frac{1+k}{n} - \frac{n-(1+k)}{n(n-1)}\right) &= \log\left(\frac{\frac{n-(1+k)}{n(n-1)}}{\frac{1+k}{n}}\right) \\ \lambda_2 &= \frac{\log\left(\frac{n-(1+k)}{(n-1)(1+k)}\right)}{2\frac{n(n-1)}{nk}} \\ &= \frac{k\log\left(\frac{n-(1+k)}{(n-1)(1+k)}\right)}{2(n-1)} \end{aligned}$$

The only cases in which this does not yield a solution for λ_2 are when $n = 1$ (which is not important, since there is only one possible distribution in this case) and when $n = 1 + k$. The latter case is equivalent to $u = 1$. Again, there is only one possible distribution in this case, so it must maximize entropy.

Appendix B

ID*

This appendix describes ID*, an incremental decision tree learning algorithm based on [Quinlan, 1986] and [Utgoff, 1988], and the synthetic test domain used for the tests described in Chapter 6.

B.1 Description of ID*

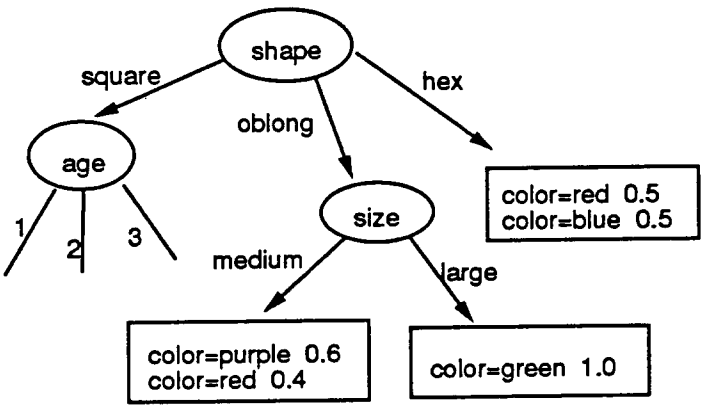

Figure B.1: Example of a decision tree

A decision tree is a tree in which each node represents a *test attribute* and each child of the node corresponds to a value of the test attribute. Each subtree covers the set of instances which matches the test attribute values along the path from the

root. Instances are stored at the leaf nodes that cover them. In Figure B.1, the test attribute at the root is "shape;" the class attribute being learned is "color." Three values for shape are represented in the tree. At the root of the subtree corresponding to "shape = square" is another internal node with test attribute "age;" this subtree has more children that are not shown. The subtree corresponding to "shape = hex" is a leaf node, containing a set of instances, half of which are red and half of which are blue.

Decision trees are used for classification by assigning a class value to each leaf node. In a probabilistic decision tree, a distribution on class values (derived from frequency counts) is stored at each leaf node. In ID*, a majority method is used for predictions: the class value of each leaf node is the class value with the highest frequency.

Quinlan [1986] describes a version of ID3 that learns probabilistic decision trees by combining an information-theoretic measure with a chi-square independence test to decide whether to split nodes in the tree (i.e., which test attributes to add when building a tree). ID5 [Utgoff, 1988] is an incremental version of ID3 (but does not learn probabilistic decision trees). In ID5, the decision tree is reshaped after each instance arrives. ID* combines both of these techniques into an incremental learner for probabilistic decision trees.

---

1. If the tree is empty, initialize it as a single leaf node containing the instance and return.
2. Insert the new instance into the tree.
3. Find the best attribute to split the tree with.
4. If the new attribute is the same as the current test attribute, recurse to each child (go to step 2).
5. Else if the new attribute is null (i.e., the tree should not be split at all), collapse the tree into a single leaf node.
6. Else pull up the new test attribute to the root.

Figure B.2: Top-level incremental learning algorithm

The top-level learning algorithm, which is called when each new instance

1. Compute the chi-square statistic for each attribute A. n is the number of instances in the subtree; $N[i,c]$ is the number of instances with the ith value of A that have class value c, and $*$ indicates summation over a parameter.

$$N'[i,c] = N[i,*] * N[*,c]/n$$

is referred to as the expectation and the chi-square statistic is

$$\chi_A = \sum_i \sum_c \frac{(N[i,c] = N'[i,c])^2}{N'[i,c]}$$

χ_A has $(k-1)*(m-1)$ degrees of freedom, where k is the number of values of the attribute A and m is the number of class values.

2. For each attribute with sufficient chi-square value, compute the information gain from splitting on that attribute. S is the set of instances in the subtree; S_i is the set of instances with the ith value of attribute A. M(S), the measure of the information in a tree, is given by

$$M(S) = \sum_i -p_i \log_2 p_i$$

where the summation ranges over the class values, and p_i is the probability of the ith class value (i.e., the frequency with which the class value appears in S). The information gain from splitting is the total information in the tree, minus the information contained in the subtrees resulting from the split:

$$I_A = M(S) - \frac{1}{n} \sum_i (N[i,*] * M(S_i))$$

3. Return the attribute from step two with highest information gain (if there are none, return null).

Figure B.3: Algorithm for determining the best test attribute to split a tree

1. If the root is a leaf node, create a subtree by splitting on A and return.
2. If the test attribute at the root is the same as A, return.
3. Recursively pull up the test attribute to the root of each subtree.
4. Swap A with R, the test attribute at the root of the tree, so that the attribute at the root of each subtree is R and the attribute at the root of the tree is A.

Figure B.4: Algorithm for pulling a test attribute A to the root of a tree

arrives, is given in Figure B.2. Step (2) inserts the instance into the appropriate leaf node in the tree. Step (3)—finding the best attribute to split the tree with—uses Quinlan's chi-square test for statistical independence, given in Figure B.3. If the chi-square value is high (above a tabulated value for a given confidence level), the value of the test attribute and the class value are unlikely to be independent. Quinlan says that

> [o]ne minor difficulty is that the chi-square test is unreliable for very small values of the expectations N', so the common practice of using the test only when all values of N' are at least 4 has been followed.

We use the test only when all values of N' are at least 5, unless the chi-square value is extremely low (less than 0.1). In other words, the attribute has "sufficient chi-square value" in step (2) if the chi-square value is greater than the tabulated 90% confidence level for a chi-square statistic with $(k-1)(m-1)$ degrees of freedom, *or* if the chi-square value is greater than 0.1 and some value of N' is less than or equal to 5.

The recursive algorithm for pulling up a new test attribute A to the root of a tree (step (6) of the top-level learning algorithm) is given in Figure B.4. This algorithm is self-explanatory except for step (4), which simply involves generating the appropriate value branches and adjusting the weights properly.

B.2 Description of the Test Domain

The synthetic test domain for the ID* tests described in Chapter 5 includes six predictive features. Table B.1 gives the names and values for each feature and the uniformity of color given each feature. The class feature (to be predicted) is "color," with four values (purple, red, blue, and green). The prior (unconditional) uniformity of color is .25 (i.e., given no other information, each value is equally likely). The predictive features are conditionally independent of each other, given color.

Table B.2 shows the table of probabilities that was built using these uniformities. The probabilities in the table represent the probability that the predictive feature takes on the specified value, given the value of color: $P(F = f_i|\text{color} = c_i)$.

Feature name (F)	$U(\text{color}\|F)$	Values
Location	.81	box1, box2, box3, box4
Shape	.58	square, hex, oblong, round
Texture	.33	smooth, rough, slimy, sticky
Size	.27	small, medium, large
Age	.27	1, 2, 3, 4, 5, 6
Smell	.25	putrid, nauseating, horrible, vile, lovely

Table B.1: Predictive features with uniformities and values

Color		purple	red	blue	green
Shape	square	.75	1/12	1/12	1/12
	round	1/12	1/12	.75	1/12
	hex	1/12	.75	1/12	1/12
	oblong	1/12	1/12	1/12	.75
Location	box1	1/30	.9	1/30	1/30
	box2	.9	1/30	1/30	1/30
	box3	1/30	1/30	.9	1/30
	box4	1/30	1/30	1/30	.9
Size	small	.3	.33	.3	.4
	medium	.3	.33	.4	.3
	large	.4	.34	.3	.3
Texture	smooth	.5	1/6	1/6	1/6
	rough	1/6	1/6	.5	1/6
	slimy	1/6	1/6	1/6	.5
	sticky	1/6	.5	1/6	1/6
Smell	putrid	.2	.2	.2	.2
	nauseating	.2	.2	.2	.2
	horrible	.2	.2	.2	.2
	vile	.2	.2	.2	.2
	lovely	.2	.2	.2	.2
Age	1	.2	.15	.16	.16
	2	.15	.2	.16	.16
	3	.15	.15	.16	.2
	4	.15	.2	.16	.16
	5	.15	.15	.2	.16
	6	.2	.15	.16	.16

Table B.2: Probability table for the synthetic domain

We next show that the probabilities satisfy the uniformity value for one of the features (age). Recall that the uniformity of color given age is equal to the probability that two randomly chosen instances have the same color, given that have the same age:

$$U(\text{color}|\text{age}) = P(c_1 = c_2 | a_1 = a_2)$$

where c_1, a_1, c_2 and a_2 represent the color and age values for two randomly chosen instances. This can be expanded using Bayes' formula and summing over individual probabilities as follows:

$$\begin{aligned} U(\text{color}|\text{age}) &= \frac{P(c_1 = c_2)\, P(a_1 = a_2 | c_1 = c_2)}{P(a_1 = a_2)} \\ &= \frac{\sum_i \sum_j P(c_j)^2 \sum_i \sum_j P(c_j) P(a_i|c_j)^2}{\sum_i P(a_i)^2} \end{aligned}$$

The probability of each class value, $P(c_i)$, is .25 (all are equally likely). The probabilities of a_i given c_i are given by the table. The probabilities of each value a_i can be found by summing over the class values:

$$P(a_i) = \sum_j P(a_i|c_j) P(c_j)$$

which is equivalent to summing the probabilities in the ith row of feature a in the probability table and multiplying by .25. The final uniformity value is:

$$\begin{aligned} U(\text{color}|\text{age}) &= \frac{4(.25)^2 .25[6(.2)^2 + 8(.15)^2 + 10(.16)^2]}{4(.67 * .25)^2 + 2(.66 * .25)^2} \\ &= \frac{.25^2(.24 + .18 + .26)}{.11 + .05} \\ &= .27 \end{aligned}$$

Representing the probabilities this way allows instances to be generated by first choosing a random value for color (according to a uniform distribution), then choosing values for each predictive feature separately. For example, if the color value is randomly chosen to be blue, the probability that the instance is in box3 is .9.

In the tests, training examples and test examples were generated independently, but the same training examples and test examples were used for every bias on a given run.

Appendix C

Generating Internal States

This appendix describes a method that allows an agent (PR) to generate internal states which represent unobservable properties of the world. The method assumes that the world is deterministic and can be described as a Markov process; the dependency of the next state of the environment on the current state can then be written as a determination (Section 5.1.2):

$$E_t \wedge A_t \succ E_{t+1} \tag{C.1}$$

E_t represents the state of the environment at time t, the present time, and A_t is the action performed by PR at time t.[1]

PR's sensory observations S include only a limited subset of E; that is,

$$S_t \subseteq E_t \tag{C.2}$$

Since PR does not have the information about the environment to build a model based directly on Equation C.1, it instead begins learning by making explicit assumptions about which rules are most likely to be useful in describing the environment. The *time contiguity assumption* tells PR initially to assume that only its most recent observations and actions are relevant to the next world state. One can think of this as a nonmonotonic belief that its sensory observations include all features of the

[1] In fact, if PR itself is a deterministic agent, and its internal state is part of the environment, then A_t is unnecessary in the determination. For our purposes, though, it will be clearer if we include A_t separately.

environment that are relevant in determining the next world state. This belief will be held and used for learning unless it leads to an inconsistency. Under this assumption, PR's first approximation to the determination given in Equation C.1 is:

$$S_t \wedge A_t \succ S_{t+1} \tag{C.3}$$

This approximation lets PR focus on those features of the environment that are most likely to be relevant to the new world state. When PR discovers that this determination does not allow it to learn a consistent world model, it relaxes the time contiguity assumption. Successive approximations to Equation C.1 have the form

$$S_{t-n,\ldots,t} \wedge A_{t-n,\ldots,t} \succ S_{t+1} \tag{C.4}$$

Note that Equation C.3 is a special case of Equation C.4 for $n = 0$. PR increases n each time it cannot find a consistent world model using the previous determination.

Letting $n = 1$ in Equation C.4 gives:

$$S_{t-1} \wedge A_{t-1} \wedge S_t \wedge A_t \succ S_{t+1} \tag{C.5}$$

Equation C.1 tells PR that S_{t-1} and A_{t-1} do not directly affect S_{t+1}, but rather affect E_t in some unperceived way (i.e., they affect $E_t \setminus S_t$). The relaxation process lets PR partially overcome its sensory limitations by "remembering" earlier observations so it can refer to properties of the environment that can no longer be observed directly. We refer to this indirect knowledge as PR's *internal states*, I_t. I_t represents the changes in E_t that PR knows have happened, because they somehow affected S_{t+1}, but that it cannot explain using only S_t and A_t.

$$S_{t-1} \wedge A_{t-1} \wedge I_{t-1} \succ I_t \tag{C.6}$$

Since all the relevant information for the determination in Equation C.5 is contained in I_t, Equation C.5 can be rewritten as

$$S_t \wedge A_t \wedge I_t \succ S_{t+1} \tag{C.7}$$

An example of an internal state being generated by this process is given in Section C.

In addition to allowing the agent to generate a consistent theory in cases where the time contiguity assumption has been found not to hold, internal states define new predicates about past events which may be relevant to rules yet to be learned. These correspond to predicates generated by *Dichotomize* (Section C.1).

Since I_t is considered to be part of PR's current state, once the predicates in I_t have been defined, the new version of the time contiguity assumption, given in Equation C.7, still holds. In other words, PR only relaxes the time contiguity assumption locally, not globally, so although it may be necessary to relax the assumption temporarily in order to learn a particular rule, PR will continue to learn other rules using the original assumption.

C.1 Dichotomization

Dichotomization is a rule transformation operator described in [Muggleton, 1987] that can also be viewed as an inverse resolution operator [Muggleton and Buntine, 1988]. *Dichotomize* takes a set of mixed positive and negative examples of a concept such as

$$X \leftarrow a \wedge b \wedge c$$
$$\neg X \leftarrow a \wedge d \wedge e$$

and generates a more general set of rules using an invented predicate p:

$$X \leftarrow a \wedge p$$
$$\neg X \leftarrow a \wedge \neg p$$
$$p \leftarrow b \wedge c$$
$$\neg p \leftarrow d \wedge e$$

The PR version of *Dichotomize* extends the definition slightly to apply to sets of rules defining any mutually exclusive concepts (rather than just positive and negative examples of one concept). An example is given in the next section.

C.2 An Example in the RALPH World

One way that PR can tell that the time contiguity assumption does not hold is if two situations that appear the same have different outcomes. (*NB* this is only true for deterministic environments.) In this case, *Dichotomize* will generate a new predicate with a logically inconsistent definition.

The behavior of PR is slightly different in this world than in the other worlds described in the thesis. PR's utility is increased every time it eats food. Before it can eat, it must have actually picked up the food using the `:grasp` action. In order to pick up the food, it must be in the same node; it can detect this as $\texttt{vision}(t, \texttt{food}, 0)$. However, once it picks the food up, it can no longer see it. For the purposes of this example, we further assume that PR does not have a `food-smell` sensor.

After a number of situations in which PR eats while grasping food, it will learn the rule

$$\texttt{action}(t, \texttt{:munch}) \rightarrow \Delta\mathrm{u}(t+1, 90)$$

If PR tries to execute the action `:munch` when it is not holding food, nothing happens. The learned rule for this case is:

$$\texttt{action}(t, \texttt{:munch}) \rightarrow \Delta\mathrm{u}(t+1, -10)$$

Δu cannot be 90 and -10 at the same time, so these rules define two mutually exclusive concepts. *Dichotomize* applies and generates the new rules

$$\begin{aligned}
\texttt{action}(t, \texttt{:munch}) \wedge \mathrm{p}(t) &\rightarrow \Delta\mathrm{u}(t+1, 90) \\
\texttt{action}(t, \texttt{:munch}) \wedge \neg\mathrm{p}(t) &\rightarrow \Delta\mathrm{u}(t+1, -10) \\
\square &\rightarrow \mathrm{p}(t) \\
\square &\rightarrow \neg\mathrm{p}(t)
\end{aligned}$$

p is intended to define whether or not eating increases utility, but given the currently available information, p's definition is inconsistent.

When this condition is detected, the relaxation process is invoked. This process involves adding the available information for the two examples at time $t-1$

and re-invoking *Dichotomize.* The new rules are

$$\texttt{action}(t, :\texttt{munch}) \wedge \mathrm{p}(t) \rightarrow \Delta\mathrm{u}(t+1, 90)$$

$$\texttt{action}(t, :\texttt{munch}) \wedge \neg\mathrm{p}(t) \rightarrow \Delta\mathrm{u}(t+1, -10)$$

$$\texttt{action}(t-1, :\texttt{grasp}) \wedge \texttt{vision}(t-1, \texttt{food}, 0) \rightarrow \mathrm{p}(t)$$

$$\neg\texttt{vision}(t-1, \texttt{food}, 0) \rightarrow \neg\mathrm{p}(t)$$

$$\neg\texttt{action}(t-1, :\texttt{grasp}) \rightarrow \neg\mathrm{p}(t)$$

p represents an internal state of "holding food." The definition of p could be generalized to apply to any object, and would then represent the more general internal state "holding" which PR may find useful in other situations (e.g., holding keys allows PR to open doors).