Chapter 1

Introduction

> Life is like playing a violin solo in public and learning the instrument as one goes on.
>
> —*Samuel Butler*

Intelligence reveals itself in many ways; one of the most impressive is the ability to adapt to unknown or changing environments. Without this capability, one can only respond to situations according to instinct, or by applying a previously-written set of rules. If our machines are to be intelligent servants, they must not require us to give them precisely-defined lists of rules for every possible contingency.

Learning "what to look for" in a new situation is one of the hardest aspects of adaptive behavior. It is also one of the critical differences between experts and novices; the experts are able to ignore irrelevant information and focus on details which have a bearing on the task at hand. For example, a first-year medical student will often reach a wrong diagnosis because he has not learned which symptoms are the important ones, and is misled by irrelevant features—"red herrings." If he looks at a blood smear, he will often miss the pathologic clue — say, a fairly rare myeloblast—because of odd shapes of the much more abundant red cells, which are of no importance (Professor Archie MacKinney, University of Wisconsin School of Medicine, personal communication). An expert chess player can quickly reproduce a chess position taken from a game after a

five-second glance at the board—but only if the pieces are placed according to an actual game. With randomly generated chess boards, the expert does as poorly as a beginner in reconstructing the board (de Groot, 1965). In both examples, the expert knows "what to look for" in order to act intelligently. The important features are those which help him perform the task at hand, even though these features may not be sufficient for some other task (such as reproducing random chess-boards).

Although knowing the important features is critical for intelligent action, it is difficult to learn an effective representation from scratch. Yet this is a critical component of adaptive problem-solving: learning to identify the important features through our interaction with the world. This dissertation attempts to show what is required for such on-line feature extraction to succeed. To answer this question in its most general form would require an analysis of representation and cognition far beyond that possible in a single dissertation. Therefore, this dissertation frames the issues within a particular framework for understanding intelligent action: reinforcement learning.

1.1 The focus of this dissertation

The central question addressed by this dissertation is this: *How can an autonomous agent construct a good representation for its task, as it learns the task?* In other words, how can the agent learn "what to look for?" This question leads to the related questions *What are the characteristics of a good representation?* and *How do these characteristics affect learning?* These are profound, open questions; the aim of this dissertation is simply to indicate new directions for the analysis and to propose some provisional criteria for constructing and evaluating representations.

Because this dissertation is about representation—how representations contribute

to problem-solving, and how we can learn them—it takes some of the other parts of the problem for granted. The analysis will assume that we learn the task by learning a set of action values, which are estimates of the long-term reward which will result from taking various actions. Much previous research has been devoted to showing how an agent can learn these values, and how it can be sure that the values converge. This dissertation builds on previous work by showing the link between representation and action values. It analyzes representations in terms of the "true" action values—the steady-state values that the agent would eventually arrive at after sufficient experience, leaving aside the questions of how these values are learned. The assumption is that a representation has certain fixed properties which will either help or hinder the agent in a particular task. Therefore, the steady-state action values reflect these properties of the representation, independent of the particular learning strategy used to learn them (assuming that it works). The agent may converge upon a set of value estimates for various actions and policies, but whether these final estimates are appropriate to the task depends critically on the representation.

This dissertation attempts to find principled answers to questions of representation. The analysis makes some basic assumptions about the task, but is meant to describe principles which are independent of any single task. Although some of the ideas are illustrated by means of a very simple gridworld task, the reader must bear in mind that the autonomous agent lacks our human understanding of grid problems and treats them the same way it treats any arbitrary task — as a black box providing it with certain sensory inputs, effector outputs, and occasional reward signals.

The remainder of this introduction presents reinforcement learning, and shows how the representation problem arises. At its conclusion, the chapter lists the dissertation's

contributions and outlines the following chapters.

1.2 Reinforcement Learning

1.2.1 A qualitative introduction

In a reinforcement learning task, an agent faces the world without a script; it must explore its environment and experiment in order to learn what to do when it faces different situations. Although it does not have a teacher, the agent does receive rewards or reinforcements from time to time. Unfortunately, there can be long delays between an action and the resulting reward; as a result, the agent faces a difficult *credit-assignment problem* in determining which action was responsible for a negative reward, and what it should have done differently. Learning an effective representation can be a profoundly difficult task in this paradigm.

Reinforcement learning is an important field of study for two reasons. First, it may lead to a better understanding of cognition in general, by providing a quantitative model for the analysis of goal-seeking behavior and intelligent adaptation. This model grounds the values of the agent's actions in the rewards it receives from its environment as a result of the actions it chooses. Second, reinforcement learning studies may show us how to make software more useful by allowing it to adapt to the needs of a task instead of having to be explicitly programmed for each narrowly-defined set of circumstances.

1.2.2 The distinguishing characteristic

The distinguishing characteristic of reinforcement learning is the type of feedback given to the agent—not the type of algorithm or representation employed by the agent. In

reinforcement learning, the feedback to the agent is usually just a number, possibly delayed in time. Thus if we categorize learning problems along a continuum, from rich feedback to sparse feedback, reinforcement learning problems are close to the sparse end of the feedback spectrum. In contrast, most supervised learning problems are at the rich end of the scale: the agent has a teacher which presents it with a series of situations to be classified, along with the correct response for each situation. This means that the agent is shown which situations are important, and it can immediately correct each action by comparing it with the correct response. One way of making the feedback less informative is to merely give the agent a number indicating how useful its response was, instead of telling it what it should have done. Another way of making the feedback more sparse is to allow it to be delayed in time, only given for certain "result" states. In reinforcement learning tasks, the agent typically has to cope with both of these complications. Therefore reinforcement learning is a kind of "trial-and-error" learning with occasional feedback from a critic, rather than from a teacher. Yet there are learning problems with even less feedback, in which the agent associates data points with action outcomes, without any external guidance at all. Although reinforcement learning systems may include mechanisms for assimilating the advice of a teacher (for example, see Maclin and Shavlik, 1996), this dissertation focuses on the representational issues that arise in the basic reinforcement learning problem.

1.2.3 Examples of reinforcement learning tasks

The practical applications of reinforcement learning are tasks where we need a system to adapt to its environment, but find it infeasible to provide the system with a teacher. For some tasks, specifying a complete list of the important situations to be learned may

simply be too tedious or expensive. In this case, it may be more economical to enable the system to detect its current state and its degree of success and failure, and then allow the system to learn to maximize its successes. Tesauro's (1995) TD-Gammon used this approach to learn to play backgammon at the level of world-class human players (but note that TD-Gammon also incorporated some helpful bias in the form of human-supplied feature detectors, which is relevant to the issue of feature extraction, discussed below).

In some cases, it might be impossible to anticipate every possible scenario the system could face. A robot explorer should be robust and flexible in its ability to handle unanticipated situations, especially if it is at a great distance, and there is a significant time lag for commands from a controlling station. This is true of the robotic exploration of our solar system, and a reason why some reinforcement learning capability could be important for such tasks.

Sometimes the optimal strategy for a task is simply unknown; in such cases, we would like the learning system to learn a strategy better than its teachers. For example, Crites and Barto (1996) applied reinforcement learning to a four-elevator, ten-floor system, and produced a control policy which out-performed the elevator motion planning schemes devised by Otis engineers. Another example is the job-shop system of Zhang and Dietterich (1996), which used reinforcement learning to produce better space shuttle pre-launch scheduling than NASA experts.

As computers, networks, and embedded computer control become more complex and more prevalent, reinforcement learning may be a key approach for flexible, robust performance without the expense of explicit programming.

1.2.4 A brief history of reinforcement learning

Modern reinforcement learning descends from two separate research threads: the study of animal learning, and the solution of optimal control problems using value functions and dynamic programming. From the research on animal learning, the field draws its emphasis on learning by trial-and-error. For example, Edward Thorndike's "Law of Effect" described how animals alter their tendency to select particular responses to a situation, according to the strength of the satisfaction or discomfort which results as the animal tries various actions in various situations (Thorndike, 1911).

From optimal control, reinforcement learning gained a mathematical formulation of tasks as Markov decision processes, a characterization of the value of a state in terms of the optimal return function (the expectation of the sum of future rewards if the agent acts optimally from this point onward), and an incremental method of calculating state values. An important distinction is that optimal control methods usually require complete knowledge of the system to be controlled, and this information is typically not available to the reinforcement learner. Instead, in a reinforcement learning task, the agent must learn the task dynamics along with the values of states, as it experiences the results of its actions.

These two research threads came together in the late 1980s, along with a third thread concerning temporal-difference learning. In temporal-difference (TD) methods, we make repeated predictions of the value of some quantity, but we update the system according to error estimates that we get by comparing each estimate with the one which followed it, instead of by comparing each estimate with the final outcome. Temporal-difference learning originated in animal learning psychology with the concept of secondary reinforcers. Examples of work incorporating TD methods include Arthur

Samuel's (1959) famous checkers program, Holland's (1986) Bucket-Brigade algorithm, the Adaptive Heuristic Critic (Sutton, 1984; Barto, Sutton & Anderson, 1983), and Richard Sutton's (1988) presentation of TD(λ) as a general prediction method.

In 1989, Christopher Watkins brought these threads together with his development of Q-learning. In Q-learning, the agent maintains a value for each (state, action) pairing, representing a prediction of the worth of taking that action from the state. (These values were represented by the function Q; hence the name "Q-learning"). The agent updates these action values according to the difference between the values at the current state and the best action value of the resulting state. Thus Q-learning combines a trial-and-error sampling of the problem space with an iterative, temporal-difference method for updating the values. The agent updates these values according to its experience of going from one state to the next, either in the real world, or by hypothetical actions in an internal model of the world.

The temporal-difference method is not the only way of doing reinforcement learning. Other methods, such as Monte Carlo, attempt to correlate actions directly with the eventual rewards without considering the intervening history of other states. What all these methods have in common is that they learn a set of action values, based on the results of the agent's actions in the task. These action values allow the agent to act intelligently by always selecting the action which has the highest potential for future reward (the *greedy policy*). This approach is called *value iteration.* Alternatively, we may learn reinforcement learning tasks by updating policy functions without learning action values; this approach is called *policy iteration.* The analysis of this dissertation is based on the value iteration approach.

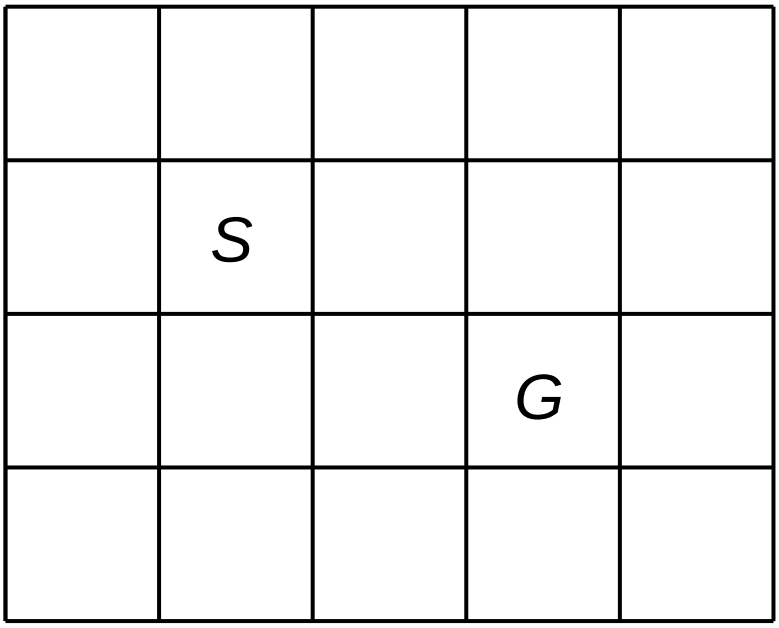

Figure 1: A 5×4 gridworld with start state S (2, 3) and goal state G (4, 2)

1.2.5 An illustration of reinforcement learning

Many of the issues of reinforcement learning, feature extraction, cognitive economy and efficient exploration may be demonstrated by a simple gridworld task. This section shows how an intelligent agent might create a more efficient representation for the task. Although the simplicity of this particular task makes such efforts unnecessary, this exercise brings to light the basic issues, which will be covered in more depth in later chapters. These issues become critical for designing good representations in complex tasks, but the simplicity of the gridworld task makes it a better introduction and demonstration.

The gridworld is shown in Figure 1. The bottom-left cell has coordinates (1, 1) in the grid, and the top-right cell is at (5, 4). In this task, the learning agent starts in state S and its goal is to enter state G as quickly as possible without hitting the walls enclosing the grid. The agent has four possible actions: left, right, up, down. It has no

idea what its goal is, apart from the rewards it receives: when the agent enters state G, it is given a positive reward of $+1$ and the episode ends; when it takes an action which moves it against the enclosing wall it receives a reward of -1, but its position remains the same. Otherwise the agent's reward is 0, and its actions move it one cell across the grid.

Since the agent has no teacher to focus its attention, it learns by stumbling along from its start state until it hits the goal state. If it is extremely lucky, its first attempt may be an optimal path to the goal, say, right, right, down, or down, right, right. More likely, the agent conducts a random walk: it retraces its steps a few times, hits the wall several times, and finally stumbles upon the goal. How can the agent discover a good policy, which produces one of the optimal paths to the goal? This is the problem of reinforcement learning. In general, the agent needs to learn the best action to take from any state it finds itself in. The standard way of doing this is for the agent to learn the action values, as it observes the results of its actions in the grid.

Notice that the goals of the problem depend on the rewards. Changing the reward function changes the nature of the task. For example, we could change the nature of the task from *goal-seeking* to *failure-avoidance* by changing the reward function and list of terminal states: Punish the agent with a reward of -1 whenever it hits one of the outer cells, and otherwise give no reward. (That is, the reward is zero for all actions which leave the agent within the "safe" interior—cells [2, 2], [3, 2], [4, 2], [2, 3], [3, 3], [4, 3]). Now the agent's objective is to avoid the exterior of the grid. If we also specify that the outer cells are terminal but that G is no longer a special terminal cell, the agent's initial episodes will usually result in a failed episode, as it eventually wanders into one of the outer cells. The optimal policy is to wander about in the interior of

the grid. Because most of the agent's initial efforts appear to lead to failure, finding important features can be especially difficult in failure-avoidance tasks. Although the agent never receives positive feedback (it can never "win"), it can eventually learn that some actions postpone failure indefinitely.

In order to know which action to take from a particular state, the agent needs to know which action will lead to the best reward in the long run. The agent can select its actions intelligently by constructing an action value function, $Q(s, a)$, which measures the "goodness" of individual actions. This function should indicate the total reward the agent will receive if it takes action a from state s, acting optimally thereafter. In addition, the function should give higher weight to rewards which come sooner rather than later.

Suppose that as a consequence of taking action a from state s, the agent experiences the reward $r(s, a)$ from the environment and ends up in state s_1. From each successive state s_i, suppose that the agent selects the action with greatest reward, which we will denote a_i^*. The sequence of rewards experienced by the agent is

$$r(s, a), r(s_1, a_1^*), r(s_2, a_2^*), r(s_3, a_3^*), \ldots$$

Let $\gamma < 1$ be a discount factor to be applied to the rewards. We define the action value $Q(s, a)$ as the sum of discounted rewards resulting from taking action a and then choosing the best possible actions for the rest of the episode:

$$Q(s, a) = r(s, a) + \gamma r(s_1, a_1^*) + \gamma^2 r(s_2, a_2^*) + \gamma^3 r(s_3, a_3^*) + \cdots$$

This sum is called a *return.* If action a is optimal, then $Q(s, a)$ is equivalent to the maximum return from s, because the remaining actions, a_i^*, are optimal.

The next chapter develops a more sophisticated definition of action values, in which the rewards and the state transitions may both be stochastic, but this simpler definition

will suffice for a demonstration. We can calculate the action values for our grid task as follows. First, calculate the maximum return possible for each cell. This is easy to calculate because of the following considerations. The only rewards in this task are $+1$ for the actions which lead to G, and -1 for the actions which result in a collision with the wall. From any cell, there is always a path which leads to G without collisions with the wall; therefore, the maximum return will always be positive. Furthermore, on an optimal path the agent will only experience reward on its final move to the goal state. Therefore, the return for an optimal path will simply be the reward $(+1)$, discounted according to its distance into the future. For example, consider the cells which border G on its top, bottom, left and right edges. Since these cells are one step away from G, the agent may move from any of them directly to G, producing a maximum return of $+1$. Next, consider the cells which are one step away from one of the cells bordering G. Since these cells are two actions away from G, their maximum return will be the reward (0) for the action to the cell bordering G, plus γ times the reward $(+1)$ for moving from the border cell to G. This adds up to a return of γ. The cells whose shortest path to G is three steps will have a maximum action value of γ^2, because the only non-zero reward will still be the final one in the path, but this time it is discounted by an extra factor of γ because it is one step further in the future. In the same way, we can calculate the maximum returns for the remaining cells, as shown in Table 1.

γ^4	γ^3	γ^2	γ	γ^2
γ^3	γ^2	γ	1	γ
γ^2	γ	1	G	1
γ^3	γ^2	γ	1	γ

Table 1: Maximum returns

Given the maximum returns, it is easy to calculate the action values for the grid.

According to our definition of $Q(s, a)$, we may rewrite the action value as the sum of the immediate reward for taking action a and γ times the maximum return for the resulting state:

$$Q(s,a) = r(s,a) + \gamma(r(s_1, a_1^*) + \gamma r(s_2, a_2^*) + \gamma^2 r(s_3, a_3^*) + \cdots) = r(s,a) + \gamma R_{s_1}^*$$

where $R_{s_1}^*$ represents the maximum return from s_1. As noted above, if action a takes the agent on an optimal path toward G, then $Q(s, a)$ is the same as the maximum return calculated for s in Table 1. But if a is a bad move, $Q(s, a)$ will be less than the maximum return for s. For example, consider the value of moving `up` from cell (4, 4), which happens to be two cells above G. This action results in a collision with the wall, and a reward of -1. After moving `up`, the agent remains in (4, 4), which has a maximum return of γ. Therefore, $Q((4,4), \texttt{up}) = -1 + \gamma(\gamma) = \gamma^2 - 1$. Table 2 lists the complete set of action values for this task. Although this table does not organize the values according to their placement in the grid, it exemplifies the idea of *discrete representation.* A discrete representation has no high-level features or state-generalization; instead, each state's values are stored separately, with no connection to the values of other states. In the literature, discrete representations are often referred to as tabular (Sutton and Barto, 1998, p. 80) or look-up table representations (Watkins and Dayan, 1992); however, note that we could also have a table look-up representation where the rows of the table refer to groups of states instead of discrete states. For this reason, I will use the term discrete representation.

This example illustrates several important facts about value iteration. First, each action value collapses the entire set of future rewards into the sum of the immediate reward for taking a particular action and the discounted value of the next state. Second, we assume that the action values do not depend on how the agent arrived at its current

s\a	up	right	down	left
(1,1)	γ^3	γ^3	$\gamma^4 - 1$	$\gamma^4 - 1$
(1,2)	γ^4	γ^2	γ^4	$\gamma^3 - 1$
(1,3)	γ^5	γ^3	γ^3	$\gamma^4 - 1$
(1,4)	$\gamma^5 - 1$	γ^4	γ^4	$\gamma^5 - 1$
(2,1)	γ^2	γ^2	$\gamma^3 - 1$	γ^4
(2,2)	γ^3	γ	γ^3	γ^3
(2,3)	γ^4	γ^2	γ^2	γ^4
(2,4)	$\gamma^4 - 1$	γ^3	γ^3	γ^5
(3,1)	γ	γ	$\gamma^2 - 1$	γ^3
(3,2)	γ^2	1	γ^2	γ^2
(3,3)	γ^3	γ	γ	γ^3
(3,4)	$\gamma^3 - 1$	γ^2	γ^2	γ^4
(4,1)	1	γ^2	$\gamma - 1$	γ^2
(4,2)	—	—	—	—
(4,3)	γ^2	γ^2	1	γ^2
(4,4)	$\gamma^2 - 1$	γ^3	γ	γ^3
(5,1)	γ	$\gamma^2 - 1$	$\gamma^2 - 1$	γ
(5,2)	γ^2	$\gamma - 1$	γ^2	1
(5,3)	γ^3	$\gamma^2 - 1$	γ	γ
(5,4)	$\gamma^3 - 1$	$\gamma^3 - 1$	γ^2	γ^2

Table 2: Tabular representation of the action values $Q(s, a)$

state. This assumption—that the identity of the current state alone is sufficient to determine its value, without consideration of the agent's history—is known as the *Markov property.* Third, the agent learns the action values for the task in reverse order. That is, the agent first learns the value of actions which result in immediate rewards, then the values of the actions which led to those states where it receives immediate rewards, then the values of actions which set the stage for those intermediate actions, and so on—from the final states of the task backwards to the starting state. Fourth, the learning is complicated by the need for the agent to explore its environment. The agent lacks our human knowledge of rectangular grids; it does not know which cells result from particular actions without trying them, and it does not even know which direction

to move toward the goal state. The agent only knows its current coordinates on the grid. As far as the agent can tell, there is always a possibility that some un-sampled path may present a "short-cut" to the goal state. For example, moving left from the start state is probably a move in the wrong direction, but could be a smart move if there were some kind of automatic transport from cell $(1, 3)$ to the goal state, $(4, 2)$. Therefore, the agent must explore the grid, even though doing so may appear wasteful in the short-term.

1.3 The representation problem

Action values can be difficult to compute because the values at a particular state depend on all possible paths through future states. As the number of states grows, the number of potential paths through the space explodes.The complexity of computing the action values increases as the square of the number of states (Kaelbling, Littman and Moore, 1996). In addition, the number of distinct values which must be maintained becomes huge; for example, we need a separate row of values for each state in Table 2. In many practical problems the number of states is larger than we would wish for the size of such a table. In particular, any continuous-valued state-space contains an infinite number of states, even if the reachable portion of the space has a small area. As a result, an agent might explore the space indefinitely without twice reaching precisely the same state.

To prevent value iteration from becoming infeasible, the representation must map the actual states of the task onto a smaller number of categories. This results in a manageable number of "generalized states." Whenever the agent needs an action value for a particular state, it references the action value for the corresponding generalized

state or states. The danger is that the values associated with the generalized states are compromises, and if the representation generalizes over states which are too dissimilar, it may lose critical information needed to solve the task. The point is to know what "dissimilar" means in the context of a particular task. By grouping similar states together, the agent no longer has to re-learn the appropriate behavior for each of them separately. Thus the agent requires fewer experiences to learn the task. From the perspective of the agent, the task appears smaller and easier to solve. The trick is to know which states can be grouped together and which distinctions can be safely ignored—and to learn to detect such relationships simply by experiencing the task.

1.3.1 Features, detectors, and generalized states

A *feature* is an attribute which the agent can use to classify its current state. The features used by the agent constitute the set of things that the agent looks for in attempting to describe and understand its situation. For example, in a program which plays chess, the program might characterize the current state in terms of features such as the number of its pieces remaining on the board, the number of the opponent's pieces, whether pieces are threatened or in position to attack, and so on. If the environment simply gives the agent the identity and location of the chess pieces, the agent may need to supplement these low-level features with a set of higher-level features which are functions of the raw inputs.

I use the term *generalized state* to describe a set of states which imply the presence of a particular feature. For example, the top row of our gridworld is a generalized state which covers the individual states (1, 4), (2, 4), (3, 4), (4, 4), and (5, 4). Sometimes it will be convenient to talk about a generalized state in terms of the set of states

which it covers, but sometimes it will be more convenient to give a rule which describes which states are included. I will call such a rule or function a feature, even though it might not be given explicitly in the representation. For example, we can define a function which outputs the value 1 for states in the top row, and otherwise outputs the value 0. This function and the description "top row of the grid" are equivalent rules for the feature "top-row." Although the agent may or may not have a particular *feature detector* which outputs a 1 for states in the top row, we can still analyze the representation in terms of this feature, even though it is not explicitly realized in the agent. Therefore, generalized states and features are both ways of describing the state generalization: generalized states do so in terms of the actual sets of states, and features do so by means of rules describing those sets.

Hence a generalized state is associated with a feature describing its members. Conversely, each feature defines a corresponding generalized state, which is the set of states recognized by that feature. Given a set of binary, non-overlapping features, these generalized states are partition regions. If the features are allowed to have continuous values and overlap, we can still talk about the recognition set for a feature as a generalized state, but with the realization that these sets will then be fuzzy sets which may overlap. The point is that the action values associated with a particular feature are shared by, or generalized over, the states in the corresponding generalized state.

In the case of the discrete representation (for example, Figure 1 and Table 2), we can think of a set of features which each correspond to a particular cell in the grid. Therefore, a discrete representation of the 5×4 gridworld has a set of 20 features. The discrete representation is fine for this task, but the method does not scale to complex tasks and is simply infeasible for continuous state-spaces. State generalization

can result in a smaller representation, in which some of the features correspond to generalized states. The action values are stored on the basis of the features, so that states detected by a particular feature share the same set of action values and tend to be grouped together as "the same kind of thing." If we allow features with continuous membership functions, states may have different degrees of membership in a particular generalized state. (Another interpretation is that the features represent *probabilities* that a particular state is a member of various generalized states). Chapter 3 defines these terms explicitly, and the following chapters use them to analyze the effects of state generalization upon action value. The purpose of state generalization is to transform the agent's problem into a smaller (and we hope, simpler) problem which has the same solution. The principal benefit of such a representation is that the agent may generalize information across similar states, requiring fewer training examples to learn the action values.

Regions of similar states

In order to decide which states should be grouped together, we need to know which states count as "the same thing" in the context of the agent's task. In the goal-seeking formulation of our gridworld task, we can identify the optimal policy for a cell as an action which has the maximum action value for that cell. Since there may be multiple actions which appear optimal, we may also describe a state in terms of its preferred action set, or *preference set.* The preference sets can be useful in categorizing compatible states. For example, in cell (1, 4) the values for actions right and down are both γ^4, but left and up have values of $\gamma^5 - 1$, so the preference set for cell (1, 4) is {right, down}. Now consider cell (2, 4), where the actions right and down have values of γ^3, the

value for left is γ^5, and the value for up is $\gamma^4 - 1$. Thus (2, 4) has the same preference set as (1, 4): {right, down}. We can define regions of cells which share the same preference sets, leading to the representation shown in Figure 2. In this representation, the six

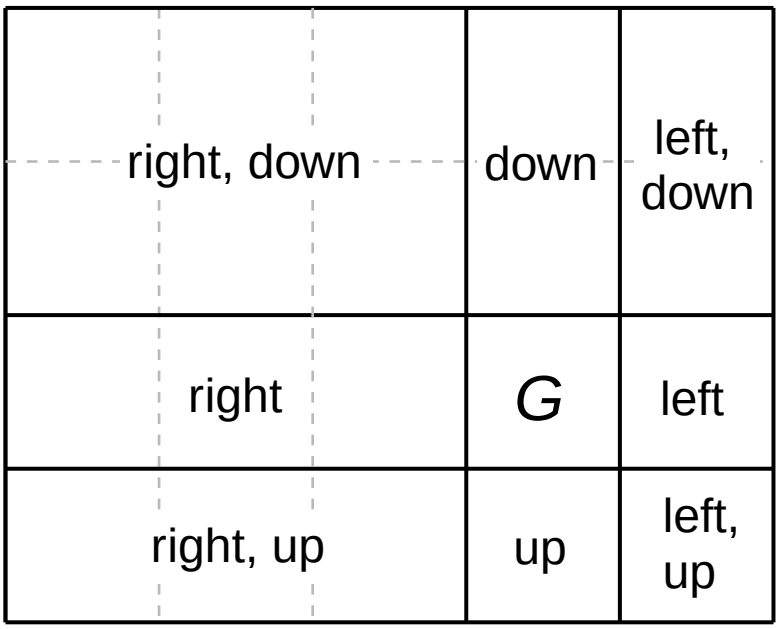

Figure 2: Gridworld partitioned according to preferred actions

cells in the upper left of the grid are grouped together because they share the same preference set: {right, down}. Therefore, we simplify the representation by collapsing them into a single state region. To the right of this group is the group of cells directly above G; the optimal policy for this group is simply to move down. We group the other cells into generalized states in the same way.

Notice that the table of maximum returns (Table 1) does not give us enough information to group states effectively because it does not take the preferred action into account. If we grouped states according to their maximum values, we would put (4, 1) and (4, 3) in the same group, even though one is below G and the other is above G. The agent needs to take different actions from these two cells, moving up from (4, 1),

and down from (4, 3). Therefore, these cells must not be grouped together.

Thus we segregate cells having different preferred actions into different state regions. We would also want to separate cells whose maximum returns differ greatly, even if they have the same preference sets—although this situation does not arise in our simple example. The distinctions between cells within the same region are irrelevant because they tell the agent nothing which will add to its ability to perform the task. For example, knowing that cell (1, 4) is distinct from (2, 4) makes no difference to the agent's policy, which is {right, down} for both these cells. Neither will the slight differences in action values for these cells result in policy differences at earlier states. We learn from this example that some distinctions are not important, but others will prove important if the agent is to perform its task well.

In this way, we may map the complete set of original states onto a smaller set of generalized states. Whether these groupings are appropriate to the task determines whether the agent will meet with success in the task. By choosing good groupings, the agent ignores irrelevant information, while still making whatever distinctions are necessary for solving the task.

How the representation affects learning

By grouping states into regions, the agent is categorizing the world it perceives. In general, the smaller the number of categories, the easier the learning task will be—provided that this categorization preserves all the information needed to perform the task. For any task, we can imagine a continuum of categorization, from very general to very specific. At one extreme, the representation is very simple, but the agent cannot perform the task because it cannot distinguish situations which require different

responses; the agent is ignoring too much information. A simple example would be a gridworld task with a single, large state region for the entire grid. At the opposite end of the continuum, the agent regards every distinct pattern of sensory information as unique; learning is very slow because the agent has to learn the same behavior over and over again for all the different states which might have been grouped together. For example, a discrete representation for a 1000×1000 cell gridworld would result in 1,000,000 cells and very slow learning of the task.

The agent can make its work easier by constructing a representation of the world which is only as fine-grained as it has to be. So the agent should adjust the granularity of the representation in order to make important distinctions where necessary, but otherwise rely on broad general categories. This may mean that the representation makes finer discriminations between states in some areas of the state space than in other areas; therefore, its resolution may vary in granularity. In addition, the representation should be tailored to the task at hand, grounded in the actual rewards that the agent experiences as a result of its actions in that task. The efficiency gained by this kind of *goal-oriented categorization* may be critical for scaling up reinforcement learning to large tasks.

1.3.2 Feature extraction

Feature extraction is the process by which the agent modifies its representation to detect features which are relevant to its task. Since any generalized state corresponds to a feature (which we can think of as a membership function for the states covered by that generalized state), state generalization and feature extraction are merely different ways of looking at the same problem. Therefore, the agent's performance depends critically

on its feature set, since the features are isomorphic to the generalized states. Future chapters will base the analysis of representation and action value in terms of features which describe the generalized states.

If the agent is unable to recognize features that distinguish states requiring different actions, the agent will not be able to perform the task. But if the features separate states which count as "the same kind of thing" in the task, the features represent irrelevant distinctions, and the agent is presented with needless complexity. Ideally, the representation should filter out irrelevant detail, while avoiding over-generalization. Furthermore, when states must be distinguished, some features may do a better job than others at making those distinctions clear; therefore, some features may appear to be more *important* than others.

How can the agent find good features as it learns the task? If we limit ourselves to small tasks which we already understand, we can design a set of feature detectors for the agent. One disadvantage to this approach is that it can involve significant human effort, which may prevent reinforcement learning from scaling up to large tasks or learning tasks which humans have not yet solved. From a research perspective, using hand-designed representations is unsatisfying because it removes the need to understand some of the most important aspects of learning.

The alternative is to automate the process of feature extraction. We want the agent to find important features on its own, grounding its decisions in the actual effects of its actions in the task. The standard approach is to approximate the action values by mathematical functions of the raw state information given by the environment; the agent then tunes these functions by gradient-descent in order to minimize the errors in its prediction of the action values. Gradient-descent techniques tune all the parameters

at once. In effect, this adjusts the value functions for regions of states, as well as the boundaries between those regions.

The gradient-descent approach is often successful, but there are reasons for suspecting that it may be ineffective for some tasks. In order for the value functions to be able to represent the values accurately, they must be tuned so that they do not generalize over states which are too dissimilar. But the error in the current action value does not indicate whether the state generalization is appropriate. For example, suppose that the value functions already generalize correctly over similar states, and put the "boundaries" in the right places. But before learning begins, the predicted values will be inaccurate because the agent has no experience with those states. The errors in the action values will lead the agent to tune the value functions, resulting in changes to the (perhaps implicit) boundaries between generalized states. This is not what was needed.

State generalization usually results in errors in the action values, but those errors are benign when the generalization is done over states which are *compatible.* Unfortunately, gradient-descent cannot distinguish benign errors from harmful ones (caused by generalization over incompatible states, and reflecting an inability to make needed distinctions in the task).

Another current approach is to attempt to ensure that features distinguish states whenever those states have significant differences in any action value. But this approach can make irrelevant distinctions if two states differ on the value of some action which is not a *preferred* action for either state. In that case, the difference is irrelevant, because the agent would never choose that action from either state. For example, in Figure 2 we grouped the cells (1, 4) and (1, 3), even though they have very different values for

the action up: $Q((1,3),\mathsf{up}) = \gamma^5$, while $Q((1,4),\mathsf{up}) = \gamma^5 - 1$. The difference in value for the action up is irrelevant, because up would be a bad action from either state, and would not be chosen.

This dissertation presents a new approach, aimed at finding features which only cover compatible states. It defines state compatibility in terms of the preference sets for the states, so that irrelevant differences are ignored. The goal is to construct a representation according to the same general principles of *cognitive economy* which enable humans and animals to cope with the cognitive demands of difficult tasks.

1.3.3 Cognitive economy and the big picture

Natural intelligences appear to represent the world in ways which filter out irrelevant information and allow them to function in a challenging environment. This seems to hold for both the organization of raw sensory information (perception) as well as higher-level constructs (conception). The resulting categorization of information allows the representation to "provide maximum information with the least cognitive effort," (Rosch, 1978, p. 28).

In the broadest sense, cognitive economy has to do with avoiding irrelevant representational distinctions. The term cognitive economy has also been used in the literature to refer to a particular strategy for information storage in semantic memory nets. To avoid confusion, this section summarizes the earlier work and explains the connection between these two different definitions of cognitive economy. Collins and Quillian (1969) used the term cognitive economy to refer to a principle for eliminating the redundant storage of information. They presented a semantic network model of human memory,

in which facts about different objects are stored at different nodes in a hierarchical network. According to Collins and Quillian, facts which are common to the instances of a category will be stored with the node for the category, rather than being redundantly stored at the nodes for each of the instances. For example, the information that birds can fly would be stored once at the node for "bird," rather than being stored at all the nodes for different kinds of birds. Information about unique traits of a particular object would be stored at the node for that object. Thus accessing category-level facts would require a movement through the hierarchy to the node for the category, resulting in longer reaction times for evaluating the truth of sentences such as "A canary can fly" than for "A canary can sing." Later work by Conrad (1972) supported the claim that words are organized hierarchically in memory, but seemed to show that attributes are stored separately with each instance, rather than once at the level of the superordinate node.

Although Conrad's work seems to be a blow against the idea of cognitive economy, it only applies to the cognitive economy of storage in semantic memory nets. It is based on a model of representation which assumes that instance and category information are stored in separate nodes, and that all possible distinctions are already present in the representation. The question then is whether information is stored in this model in such a way that storage space is conserved at the expense of processing time. Thus Conrad's results do not argue against the more general idea that representing the world efficiently makes us effective in cognitive tasks. In this dissertation, 'cognitive economy' will refer to the more general idea of making relevant distinctions while ignoring irrelevant information.

The phenomenon of *categorical perception* (CP) appears to be an example of this

type of cognitive economy. Stevan Harnad describes categorical perception as "a qualitative difference in how similar things look or sound depending on whether or not they are in the same category" (Harnad, 1987, p. 2). For example, even though color stimuli vary along a smooth continuum of wavelengths, humans break that continuum up into a small set of labelled regions.

> Physically, the wavelength spectrum varies continuously — one wave-length differs from another by simple quantitative change. Psychophysically, human observers can discriminate many wavelengths — our powers to discern are keen. Psychologically, however, hues vary in a categorical fashion—our perceptions cross more or less discretely from one wavelength region to another. Considering hue, brightness, and saturation together, we can tell literally thousands of color nuances apart, but we still partition the color space into relatively few distinct qualitative sensations. (Bornstein, 1987, p. 288)

According to Berlin and Kay (1969), humans break the color spectrum into 11 basic color categories, although languages differ in whether they have basic color terms for all 11. We judge differences between two wavelengths as smaller if they come from the same category, and larger if they come from different categories. For example, we might judge two shades of green as more similar to each other than one of those greens is to a particular shade of yellow, even though the green-yellow comparison may have an equivalent difference in wavelength.

The same effect characterizes perception of speech sounds such as the stop-consonant categories /ba/, /da/, and /ga/. "In other words, in CP there is a quantitative discontinuity in discrimination at the category boundaries of a physical continuum, as

measured by a peak in discriminative acuity at the transition region for the identification of members of adjacent categories" (Harnad, 1987, p. 3). The advantage of this kind of categorization is cognitive economy: "It is more efficient to organize the world into a small number of superordinate units than to deal with each individual exemplar" (Snowdon, 1987, p. 336).

This categorization appears to be the product of both biology and experience. That different human cultures arrive at the same categorical perception of color (Berlin and Kay, 1969) appears to show that some categories are the result of biological constraints. But other evidence indicates that perceptual categories are also the result of experience and learning, resulting in a set of perceptual distinctions which are relevant and diagnostic in the particular cognitive tasks we face. For example, infants that grow up in a particular language environment, say, English, appear to lose the ability to discriminate speech sounds absent in that environment within the first year of life. Werker and Tees (1984) chose two phonetic contrasts which are not present in English, and showed that they could be discriminated by infants from English-speaking homes at six months of age but failed to be discriminated by virtually all of those infants at 12 months. Whether these changes are permanent or not, they are evidence that human perception is organized around distinctions which give us a functional advantage in interacting with our world.

In higher-level tasks, experts appear to learn to represent the relevant details of the environment much more efficiently than novices. In his experiments with chess players, Adriaan de Groot found that the primary difference between master chess players and lesser players was that the masters were able to immediately recognize the important attributes of a chess position. Of the chessmaster, de Groot writes "as a result of his

experience he quite literally 'sees' the position in a totally different (and much more adequate) way than a weaker player [...] His extremely extensive, widely branched and highly organized system of knowledge and experience enables him, first, to recognize immediately a chess position as one belonging to an unwritten category (type) with corresponding board means to be applied, and second, to 'see' immediately and in a highly adequate way its specific, individual features against the background of the type (category)" (de Groot, 1965, p. 306). Thus "[t]he difference in achievement between master and non-master rests primarily on the fact that the master, basing himself on an enormous experience, can start his operational thinking at a much more advanced stage and can consequently function much more specifically and efficiently in his problem solving field" (ibid, p. 307).

To summarize, natural intelligences categorize information in order to reduce the cognitive load of difficult tasks. This categorization is functional, allowing us to ignore irrelevant detail, and to more easily recognize distinctions which are relevant in our world. "It is to the organism's advantage not to differentiate one stimulus from others when that differentiation is irrelevant to the purposes at hand," (Rosch, 1978, p. 29). As a result, the organism is able to perform difficult tasks in spite of its limited cognitive resources.

This dissertation takes the position that artificial intelligences should make use of similar strategies, in order to make the most efficient use of their limited cognitive resources. Specifically, feature extraction should be guided by the principle of cognitive economy. Therefore, a good representation should *not* make all possible distinctions for the tasks we face, but should use broader categories when distinctions are irrelevant to the tasks and rewards we experience. But what does it mean for a feature to be

relevant, and how does that depend on the tasks we want to solve and the nature of our environment? How does our representation change the nature of the apparent task? When should we regard things as "the same"? And how can we learn a good representation of the world through our interactions with it?

1.4 Contributions of This Dissertation

This dissertation translates the idea of cognitive economy into algorithmic criteria for feature extraction in reinforcement learning. To do this, it develops mathematical definitions of feature importance, sound decisions, state compatibility, and necessary distinctions, in terms of the rewards experienced by the agent in its task. It analyzes the connection between the representation and the resulting action values, and offers criteria which ensure that a representation is adequate for the learning of the task. These ideas are explored through theoretical analysis, as well as by implementation and simulation. Contributions include the following:

1. Characterization of learnability in terms of criteria for the allowable loss of reward at each state (*incremental regret*).

2. Definition of feature importance (relevance) in terms of action values, and analysis of the relation between importance and the robustness and efficiency of learning.

3. Characterization of necessary distinctions in terms of learnability.

4. Definition of a set of criteria for state compatibility, and demonstration of the link between these criteria and learnability. (Under certain reasonable conditions, representations which separate incompatible states are proven to meet the

learnability criteria). The compatibility rules specify when states should be considered as "the same kind of thing" in a particular task.

5. Characterization of generalized action values for features and for sets of states, including the role of our prior assumptions in choosing a definition of generalized action values. By defining these values by the steady-state expectations of reward, we may consider the generalized action values to be properties of the representation, apart from any consideration of the convergence of the learning algorithm. Therefore, these definitions of generalized action values show how the representation changes the task perceived by the agent.

6. Successful implementation of an algorithm which generates an effective representation for its task, as it goes about learning the task.

These topics are central to effective learning, because they have to do with the way that state generalization changes the agent's view of the task, whether this makes learning easier or harder, and how the agent may exploit this information to find important features and effective representations, making the most of its limited cognitive resources.

1.5 A Brief Preview of Subsequent Chapters

Chapter 2 formulates the reinforcement learning problem, establishing its essential elements and their mathematical foundation. This provides a basis for considering learning and feature extraction in terms of action values. Along the way, the chapter reviews related work.

Chapter 3 formalizes the principle of cognitive economy into objective criteria expressed in terms of action values. The chapter begins by briefly reviewing cognitive economy from the perspectives of psychologists, economists, and other reinforcement learning researchers. Then it presents the representational model, assumptions, and definitions that form the basis for the rest of the dissertation. Most of the chapter is devoted to the development of criteria for feature importance, sound decisions, necessary distinctions, representational adequacy, and state compatibility.

Chapter 4 extends the definition of action value to generalized states, including those which correspond to continuous-valued, overlapping feature detectors. The chapter indicates how our assumptions about the reinforcement learning problem can lead to different definitions of generalized action values. It applies these definitions to a simple gridworld task, and shows how state generalization changes the agent's perception of the values of its actions.

Chapter 5 analyzes the role that representation plays in allowing the agent to make sound decisions. The chapter builds a link between the previously-derived criteria for compatible states and representational adequacy: partition representations which separate incompatible states are proved to make all necessary distinctions required for the agent to solve the task. This result adds support to the conclusion that the criteria are a well-founded, principled application of cognitive economy to the reinforcement learning problem.

Chapter 6 presents an algorithm which applies the ideas developed in the previous chapters to the solution of two reinforcement learning problems: the puck-on-a-hill, and pole-balancing. The success of the algorithm in constructing an effective representation from scratch demonstrates the potential of a feature extraction strategy which is driven

by considerations of cognitive economy, rather than by the minimization of the top-down errors in action values.

Chapter 7 summarizes the dissertation and gives some thoughts about future work. The Afterword offers a brief consideration of the place of this work within the larger picture of intelligent action.