**The Markov Decision Problem**

SYSTEM: Sensors 🡪 States 🡪 Actions

P(action) < 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | +1 |
|  | WALL |  | **-1** |
|  | Start |  |  |

OBJECTIVE: Reach +1 from START without passing through location **-1**

**QUESTION: How?**

**ANSWER: Find the shortest path from START to +1 that avoids -1.**

This is a **deterministic transition model**.

**Probabilistic Model (non-deterministic)**

Move from deterministic to a non-deterministic (probabilistic) model: define a probability distribution on the directions of move (actions): North, South, East, West.

For example, suppose that each non-edge cell has the following probability distribution:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Action: move | **North** | **East** | **South** | **West** |
| Probability | 0.8 | 0.1 | 0.1 | 0 |

That is,

So, if we chose to guide the move by this probability distribution, then the move will always be to North.

But what if there is a WALL (as it is for START) to the North? Moving North is NOT an option.

Consider other cases:

1. Move East

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | +1 |  |  |  |  |  | +1 |  |  |  |  |  | +1 |
|  |  |  | -1 |  |  |  |  |  | -1 |  |  |  |  |  | **-1** |
|  | START |  |  |  |  |  | START |  |  |  |  |  | START |  |  |
| P (East | START) = 0.1 | | | |  |  | P(East,East|START)=0.01 | | | |  |  | P(North,East,East|START)=0.008 | | | |

1. Move West

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | +1 |  |  |  |  |  | +1 |  |  |  |  |  | +1 |  |  |  |  |  | +1 |
|  |  |  | -1 |  |  |  |  |  | -1 |  |  |  |  |  | **-1** | …... | |  |  |  | **-1** |
|  | START |  |  |  |  |  | START |  |  |  |  |  | START |  |  |  |  |  |  |  |  |
| P (West|START)=0.1 | | | |  |  | P(North,West|START)=0.08 | | | |  |  | P(North,North,West|START)=0.064 | | | |  |  | P(+1, …|S) = 0.00064 | | | |

P(land in +1) = 0.00064; P(land in -1) =0.000064

In general, we can illustrate ALL the moves/paths with their probabilities by trees.

Longer path to reach +1

In the Deterministic Transition Model, we use the shortest path (e.g., A\*-algorithm, Dijkstra).

**UTILITY**

We now introduce the notion of utility:

which maps the current state into the goal state,

We want the utility to be inverse related to the distance to the goal state, i.e., the smaller the distance, the higher the utility. That is

The Deterministic Model assumes a perfect world.

**From Utility to Policy**

A policy maps states into actions

**Objective**: Find the “Best Policy”

|  |
| --- |
| **The Markov Decision Problem** |
| Given a stochastic environment (i.e., governed by a matrix of transition probabilities), with a known transition model, compute the optimal policy |

Markov Property: Si, i=0,1,2… is a sequence of states

**P(Si | Si-1, Si-2, …, S0) = P(Si | Si-1)**

**Therefore, NOT every decision problem is a Markov Decision Problem (MDP).**

Now, let us formulate the MDP assuming that we can define the following:

***Pa*(*i,j*)**: probability to reach state ***j*** from state ***i*** by action ***a***

***U*(*j*)*:*** utility of state ***j***

Then, for a collection of nodes (states), **J**, and actions **A**,

That is, the optimal policy from state , is the action which maximizes the expected value of the utility over all states , that can be reached directly from state .

But in general, we do not know U!!!

**To compute U, we need to look at the entire tree of states.**

For this, we want to estimate how good a state is. So, we are going to reason as follows:

1. The utility of a state is as good as the utilities of its successor states are.

Thus, without loss of generality, if we denote states by

|  |
| --- |
| S0 S1 … Sn |
| 0 1 … n |
|  |

1. **Separable Utility**: We decompose the utility into two components:
2. Local component: REWARD
3. Non-local/successor component: Utility of successor states

Therefore, we can write

or, in general,

or, now, using the relation between Utility and the transition probability matrix

|  |
| --- |
| **(1)** |

**NOTE:**

**the local component of , DOES NOT depend on the successor states. In a way, it conveys what we mean when we say “so far so good”.**

**Equation (1) captures the OPTIMALITY PRINCIPLE from Dynamic Programming, which can be stated as follows:**

***”An optimal policy has the property that whatever the initial state and initial actions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first action.”***

For an n-step decision problem:

1. Naïve solution: ; A is the set of actions
2. DP solution: ; S: set of states

What is the “correct” value for n?

Computing the Utility function

1. Iterative Computation: **Ut(i) :**

Then define the optimal utility as the limit:

Operationally, letting be a threshold, t, a time moment, we write

1. **IF THEN *UOptimal = Ut+1***
2. **ELSEIF** ***Ut+1***< **THEN** **Stop**

**The Value Iteration Algorithm**

|  |
| --- |
| Given  , a transition matrix,  R: a reward function,  : a threshold   1. Initialize U, U1 to R 2. Repeat   U = U1  For i=1: n   * + 1. U(i) = R(i) + Max*a* Σ*j* P*a*(i,j)U(j)   End  Until |U – U1| <  Return U |

Original example revisited

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Transition Prob: P(i,j) | | |  |  | U, R matrix | | |
|  | 0.8 |  |  |  |  | U=10 |  |
| 0.1 |  | 0.1 |  |  | U=5 | R=1 | U=-8 |
|  | 0 |  |  |  |  | U=1 |  |

*a* ∈ {N, W, E, S}

Thus,

That is, the utility for center is 8.7 (going up). We can now compute the optimal policy

**ISSUES:** How to measure convergence?

Define Root Mean Square Error (RMSE)

and use to stop when RMSE(U, U1) <

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | +1 |  |  | 0.81 | 0.86 | 0.91 | +1 |
|  | Wall |  | **-1** |  |  | 0.76 | Wall | 0.65 | **-1** |
|  |  |  |  |  |  | 0.70 | 0.66 | 0.611 | 0.38 |
| Original Environment | | | |  |  | Calculate Utilities: Uoptimal | | | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | +1 |  |  |  |  |  | +1 |
|  | Wall |  | **-1** |  |  |  | Wall |  | **-1** |
|  |  |  |  |  |  |  |  |  |  |
|  | | | |  |  | Execute actions | | | |

Let g = 1, and r = -0.04 for transition to nonterminal states.

Bellman equation:

The 4 (columns) x 3(rows) grid example: Let g = 1, and r = -0.04 for transition to nonterminal states.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 | Start |  |  |  |
|  | 1 | 2 | 3 | 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Action: move | **North** | **East** | **South** | **West** |
| Probability | 0.8 | 0.1 | 0.1 | 0 |

U(1,1) =

= max { [0.8(-0.04 + g U(1,2)) + 0.1(-0.04 + g U(2,1)) + 0.1(-0.04 + g U(1,1)))], Up

[0.9(-0.04 + g U(1,1)) + 0.1(-0.04 + g U(1,2))], Left

[0.9(-0.04 + g U(1,1)) + 0.1(-0.04 + g U(2,1))], Down

[0.8(-0.04 + g U(2,1)) + 0.1(-0.04 + g U(1,2)) + 0.1(-0.04 + g U(1,1))) Right

}

=