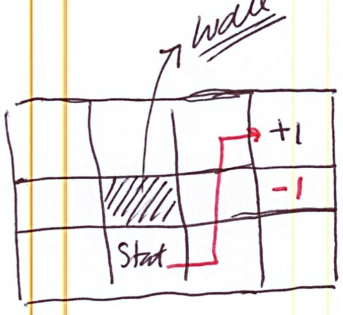
SYSTEM

Sensors — states — actions

Prob (action) < 1

Example



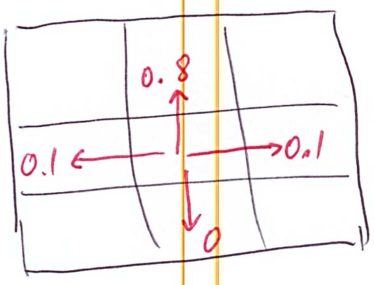
Objective

Reach +1 from START w/o passing through -1

Answer : shortest path that avoids -1

This is a deterministic transition model

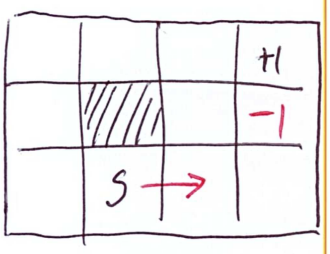
Move from deterministic to a non-deterministic (probabilistic) model: for each cell assign a probability to move N/E/S/W



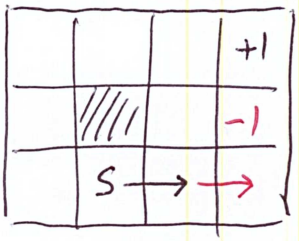
Now P(N|Start) is max and, after we choose to guide more by prob. the more will be North, where there is a wall! So, it is NOT a good option!

Let us consider the other two options.

(1) Move East



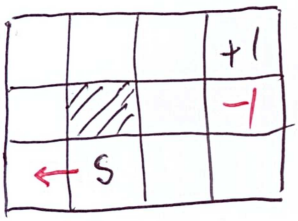
$p=0.1$



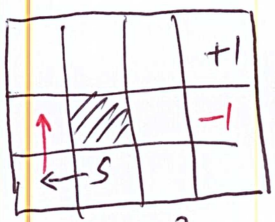
$p=0.1$

with prob $0.1 \times 0.1 \times 0.8 = 0.008$ the ~~robot~~ lands in -1 ✗

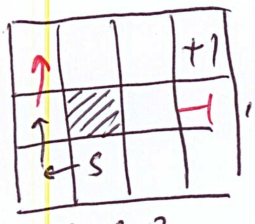
(2) Move West

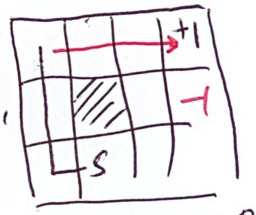


$p=0.1$



$p=0.8$





$p=0.8$ ... $0.1 \times 0.1 \times 0.1$

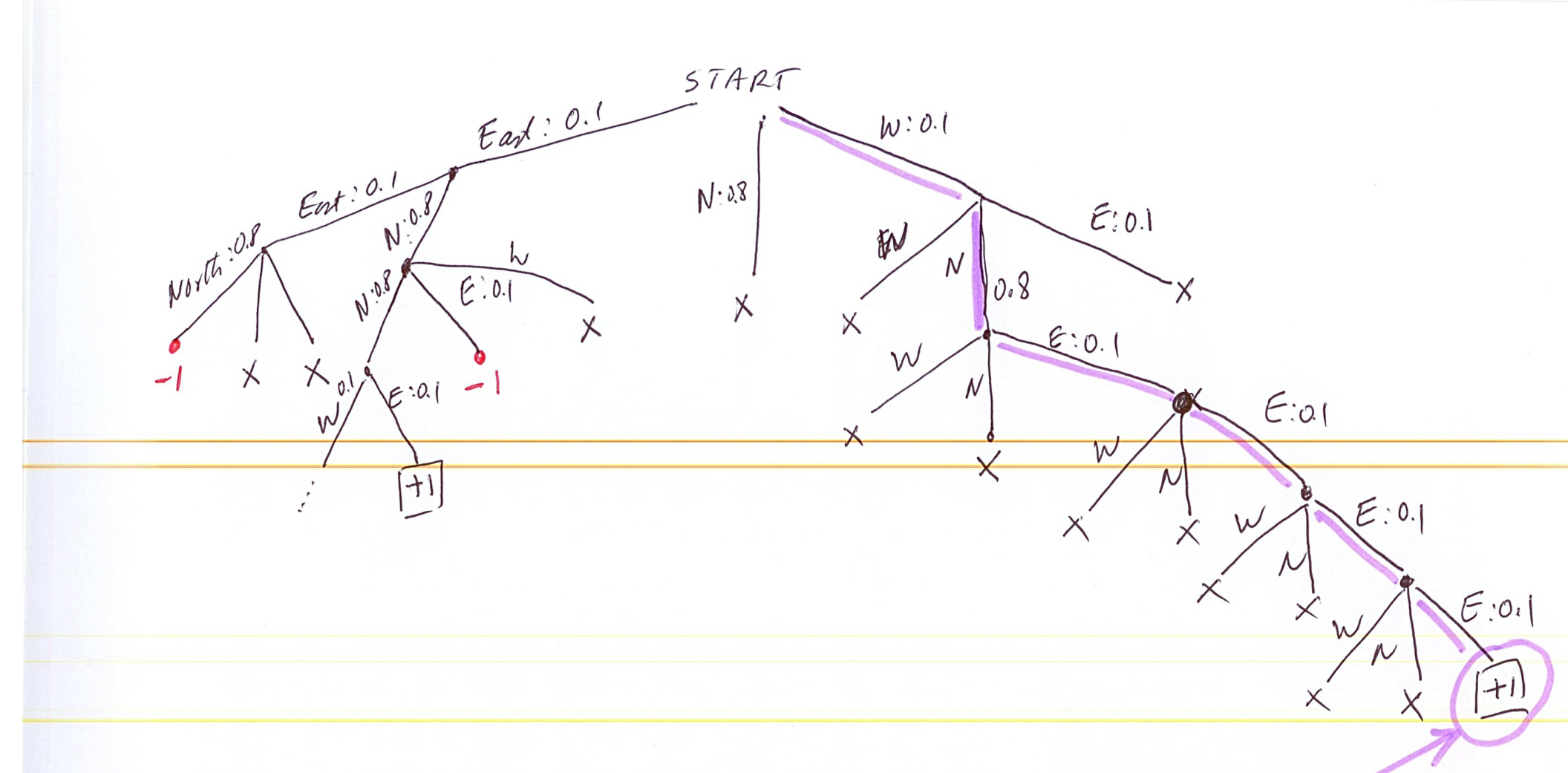
$\Rightarrow$ Prob : $0.1 \times 0.8 \times 0.8 \times 0.1 \times 0.1 \times 0.1$ to land $+1$

$) \times 0.1$ to land at -1

(

In general,

we can illustrate all the moves / paths with their probabilities by trees!



longer path to reach +1

(Highest Utility)

But what is utility?

In the Deterministic Transition Model

We use the shortest path (A*-alg, Dijkstra)

Utility is a function from the current state (a node in the graph of states) to the goal state.

We want the utility to be inverse related to the goal state : smallest distance ≡ highest utility

$$U(\text{node}) = \frac{1}{\text{dist}(\text{node}, \text{goal})}$$

The DT Model assumes a perfect world

From utility to POLICY : mapping from States to Actions

Policy : States ⟶ Actions / π is used often to denote Policy

Objective : Find "BEST POLICY"

Markov Decision Problem : Given a stochastic environment with transition model known, compute the optimal policy

Markov Property

S_i $i=0,1,\ldots$ is a sequence of states.

MP: $\boxed{P(S_i \mid S_{i-1}, \ldots S_0) = P(S_i \mid S_{i-1})}$

NOT EVERY DECISION PROBLEM IS A MDP

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

$P_a(i,j)$: prob. to reach state j from state i by action a

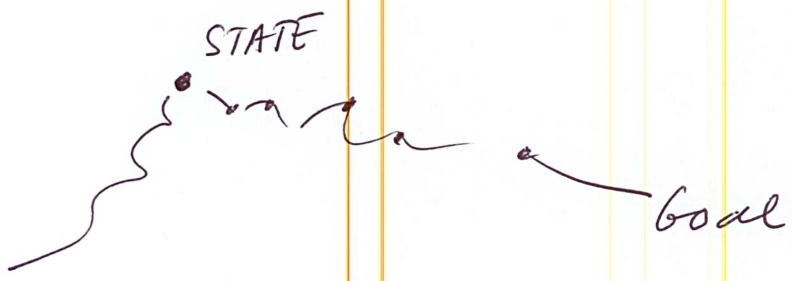
$U(j)$: utility of node/state j

Then for a collection of nodes $j \in J$, and actions $a \in A$

$$\pi_{opt}(i) = \arg\max_{a \in A} \sum_j P_a(i,j)\, U(j)$$

In general we do NOT know U!

To compute U we need to look at the tree of states. We want to estimate how good a state is.



Obviously, utility of STATE is as good as the utilities of its successor states are.

$$S_0 \rightarrow S_1 \rightarrow \cdots \rightarrow S_n$$

$$0 \rightarrow 1 \rightarrow 2 \rightarrow \cdots \rightarrow n$$

$$U(i) = f(U(i+1), \ldots U(n)) \quad f \uparrow$$

Separable utility: utility can be decomposed into two components:

- local component: REWARD
- non-local / successor component: utility of successors

$$U([0,1,\ldots,n]) = R(0) + U([1,\ldots,n])$$

In general,

$$U([i,i+1,\ldots,n]) = R(i) + U([i+1,\ldots,n])$$

or,

$$\boxed{U(i) = R(i) + \max_a \sum_j P_a(i,j)\, U(j)} \quad (1)$$

Note $R(i)$, the local component of $U(i)$ does not depend on the successor states. In a way it captures what we mean when we say "so far so good", or "easy does it".

Equation (1) captures the OPTIMALITY PRINCIPLE from Dynamic Programming.

"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:

Naive solution : $O(|A|^n)$ A: set of actions.

DP solution $O(n|A||S|)$ S: set of states

What is the "correct" value of n?

COMPUTING the utility function

IDEA FOR ITERATIVE COMPUTATION: $U_t(i)$ utility at time t

$$U_{t+1}(i) = R(i) + \max_{a \in A} \sum_j P_a(i,j)\, U_t(j)$$

Then define the optimal utility

$$U_{opt} = \lim_{t \to \infty} U_t$$

Operationally, ~~U_{t+1}~~

informally
- $U_{opt} = U_t$
- if $U_{t+1} > U_t$
  - $U_{opt} = U_{t+1}$
- else if $U_{t+1} <$ threshold
  - STOP

Formally, we have

Value Iteration Alg

Given $(P(i,j))_{i,j}$: a transition matrix

R : a reward function ; ε : threshold

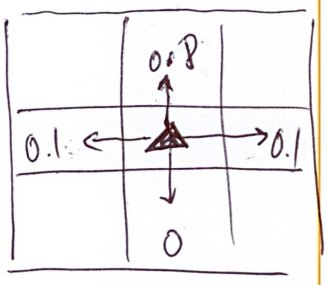
we calculate the utility as follows

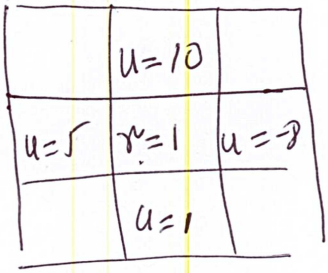
Initialize U, U_1 to R

Repeat
$U = U_1$
for $i = 1 : n$
$$U_1(i) = R(i) + \max_a \sum_j P_a(i,j)\, U(j)$$
end
until $|U - U_1| < \varepsilon$

Return U

Return to example to compute U





u, R

$a \in \{N, W, E, S\}$

a: W ← $0.1 \times 1 + \boxed{0.8 \times 5} + 0.1 \times 10 = 5.1$

a: N ↑ $0.1 \times 5 + \boxed{0.8 \times 10} + 0.1(-8) = 7.7$

a: E → $0.1 \times 1 + 0.1 \times 10 + \boxed{0.8(-8)} = -5.3$

a: S ↓ $0.1(-8) + 0.1 \times 5 + \boxed{(0.8)(1)} = 0.5$

$U(\text{center}) = R(\text{center}) + \text{Max}\{5.1(W), 7.7(N), (-5.3)(E), 0.5(S)\}$

$= 1 + 7.7(N) = 8.7(N)$

Utility for center is 8.7 (going up).

We can now compute the optimal policy

$$\pi_{opt}(i) = \arg\max_a \sum_j P_a(i,j)\, U_{opt}(j)$$

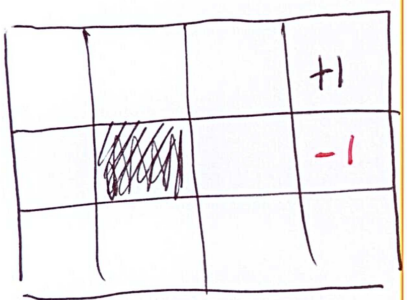
This is the optimal solution for MDP.

ISSUES: How to measure the convergence?

Use some measure such as Root Mean Square Error

$$RMS = \frac{1}{|S|}\left(\sum_{i=1}^{|S|} (U(i) - U'(i))^2\right)^{1/2}$$

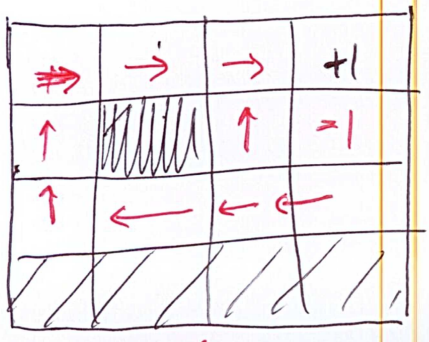
and use ε: Stop when $RMS(U, U') < \varepsilon$



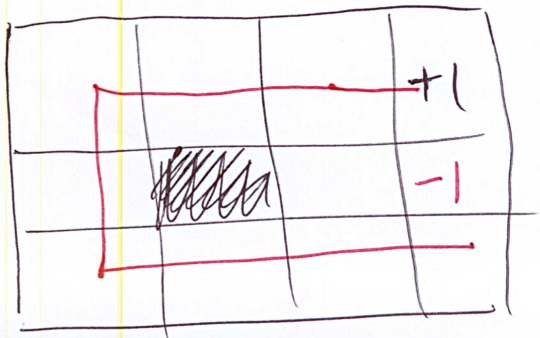
ORIGINAL ENV.

0.81	0.86	0.91	+1
0.76		0.66	-1
0.70	0.65	0.611	0.388

Calculate utilities: U_{opt}



π_{opt}



execute actions