# Machine, Data and Learning

# Decision Theory

## (How to make decisions)

Decision Theory

= Probability theory + Utility Theory

*(deals with chance)* *(deals with outcomes)*

- *Fundamental idea:*
  - The **MEU** (Maximum expected utility) principle
  - Agent is **rational** if and only if it chooses the action that yields the highest expected utility, averaged over all possible outcomes of the action
  - Weigh the utility of each outcome by the probability that it occurs

# Revisiting Romania example

- If plan1 and plan2 are the two plans:
  - Plan 1 uses route 1
    - P(home-early|plan1) = .8, while P(stuck1|plan1) = .2
    - Route 1 will be quick if flowing, but stuck for 1 hour if slow
    - ➢ U(home-early) = 100, U(stuck1) = -1000
    - ➢ Assigned numerical values to outcomes!
  - Plan 2 uses route 2
    - P(home-somewhat-early|plan2) = .7, P(stuck2|plan2) = .3
    - Route 2 will be somewhat quick if flowing, but not bad even if slow
    - ➢ U(home-somewhat-early) = 50, U(stuck2) = -10

# Application of MEU Principle

- EU(Plan1) = P(home-early | plan1) *U(home-early)
  + P(stuck1| plan1) * U(stuck1)
  = *0.8 * 100 + 0.2 * -1000 = -120*

- EU(Plan2) = P(home-somewhat-early | plan2) *U(home-somewhat-early)
  + P(stuck2 | plan2) * U(stuck2)
  *= 0.7 * 50 + 0.3 * -10 = 32*

EU (plan2) is higher, so choose plan2

# Lottery Example

- Suppose an agent  gives you a choice:
  - **Choice 1:** You will get $1,000,000
  - **Choice 2:** The agent will toss a coin
    - If heads, then you win $3,000,000
    - If tails, then you get nothing

- Simple expected utility calculations give:
  - EU(Choice1) = $1,000,000
  - EU(Choice2) = $1,500,000

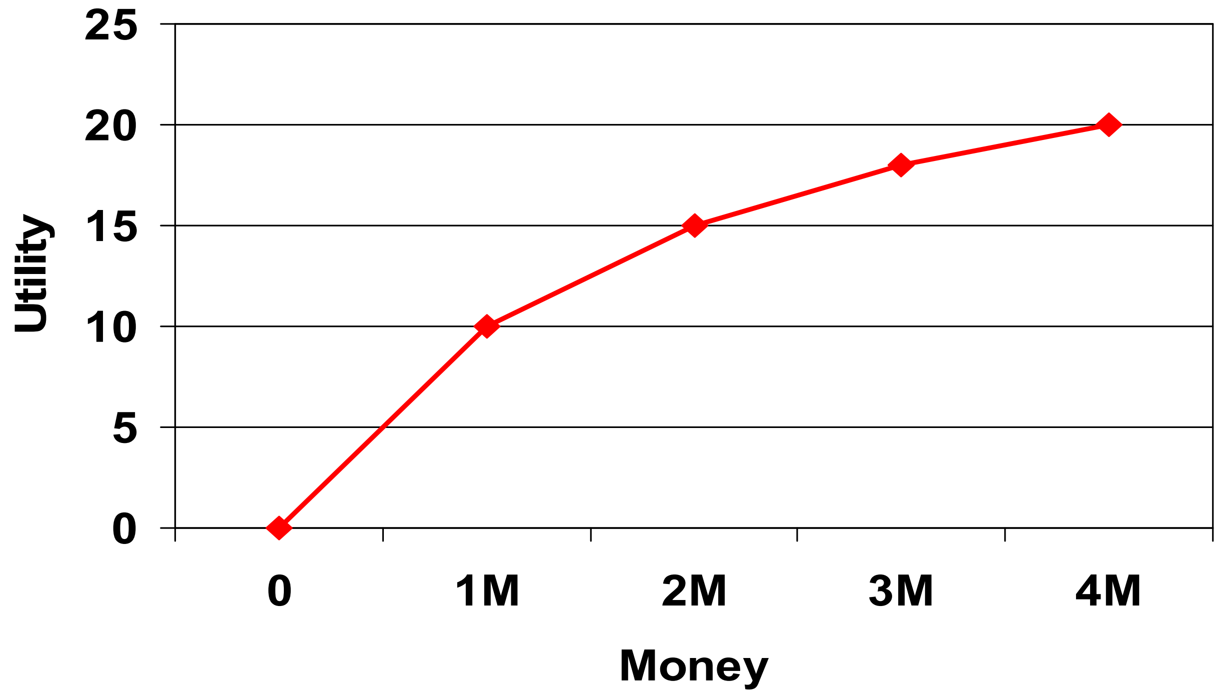
- So why did we prefer the first choice?

# Risk Aversion

- We are **risk averse**
- Our utility functions for money are as follows (!!):
  - Our first million means a lot U($1M) = 10
  - Second million not so much U($2M) = 15 (NOT 20)
  - Third million even less so U($3M) = 18 (NOT 30)
  - ….

- Additional money is not buying us as much utility
- If we plot amount of money on the x-axis and utility on the y-axis, we get a concave curve

# Answer: Risk Aversion



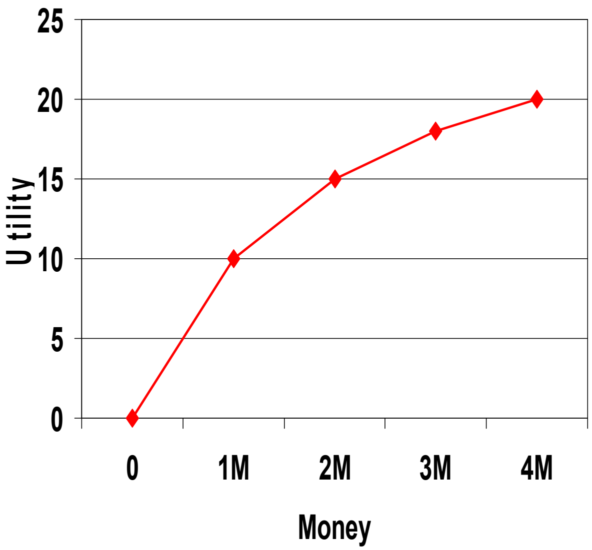

- EU(choice1) = U($1M) = 10
- EU(choice2) = 0.5*U(0) + 0.5*U($3M =18)   = 9
- That is why we prefer the sure $1M

# More Risk Aversion

- Key: Slope of utility function is **continuously decreasing**
  - We will refuse to play a monetarily fair bet
- Suppose we start with x dollars
  - We are offered a game:
    - 0.5 chance to win 1000 dollars ($c = 1000$)
    - 0.5 chance to lose 1000 dollars ($c = 1000$)
    - Expected monetary gain or loss is zero (hence monetarily fair)
  - Should be neutral to it, but seems we are not! Why?
    - $U(x + c) – U(x) < U(x) – U(x – c)$
    - $U(x + c) + U(x – c) < 2\ U(x)$
    - $[U(x + c) + U(x – c) / 2] < U(x)$
    - EU (playing the game) < EU (not playing the game)

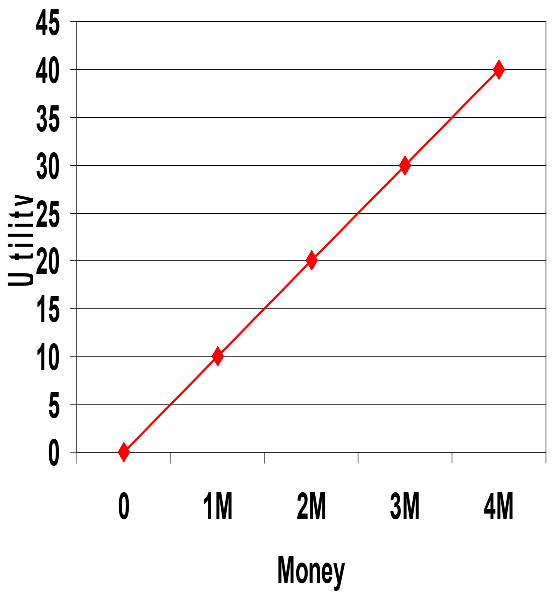
# Risk Averse, Risk Neutral
# Risk Seeking

RISK AVERSE



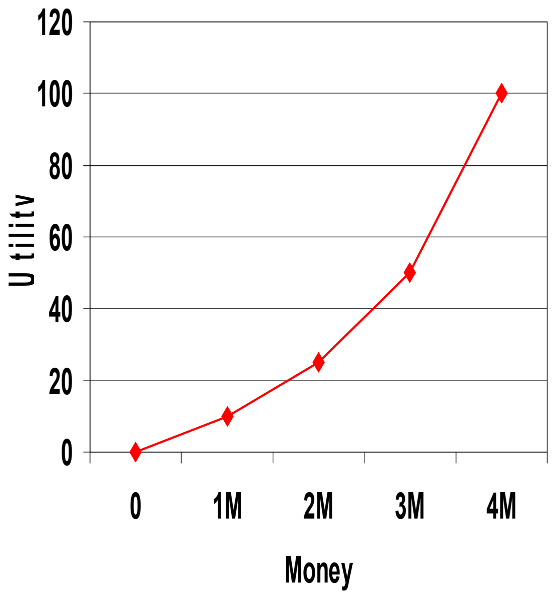

EU(Choice1) = 10
EU(Choice2) = 9

RISK NEUTRAL




EU(Choice1) =10
EU(Choice2) =15

RISK SEEKER




EU(Choice1)=10
EU(Choice2)=25

# Multiattribute Utility Theory

- Can we capture desirability of outcomes in a single utility function?
- Suppose renting an apartment
  - House1: closer-to-university, newer, costs 100 units
  - House2: Farther-from-university, older, costs 85 units
  - (Assume, you can afford up to 100 units)

- Outcomes characterized by two or more attributes
  - Attributes: X1, X2, …XN, e.g., <distance-to-univ, old/new, cost>
  - Values: x1,x2…xN,
    - Closer-to-univ = 1, farther-from-univ = 0; new = 1, old = 0
  - Apartment1: <1,1,-100>    Apartment2: <0,0, -85>
  - Which  is a better apartment? (Pairwise comparison fails)

# Multiattribute Utility Theory

- Don't get a single number, but vector of values as outcomes, <x,y>
- How do you compare values now?
  - Compare <1,1,-100> with <0,0, -85>
  - Compare <3,3,5> with <5,3,3>
- One approach is dominance (strict, stochastic…):
  - If you are lucky, find <3,3,3> and <3,3,5>
  - Values in one vector dominate values in the other vector

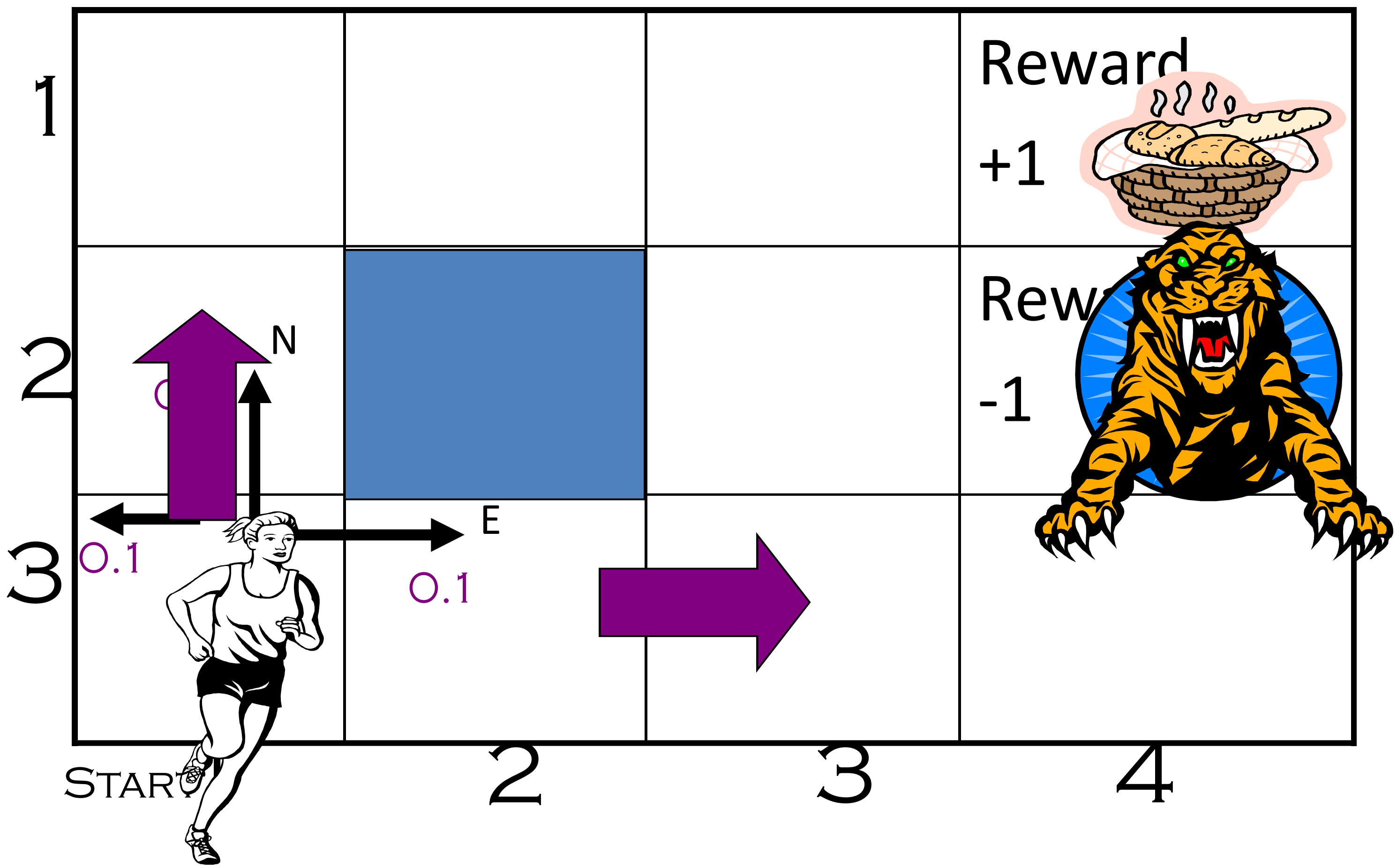
# Markov Decision Process (MDP)
# Chapter 17: Making Complex Decisions

- Defined as a tuple: <S, A, P, R>
  - **S:** State
  - **A:** Action
  - **P:** Transition function
    - Table P(s' | s, a), prob of s' given action "a" in state "s"
  - **R:** Reward
    - R(s, a) = cost or reward of taking action a in state s

- Choose a sequence of actions (not just one action)
  - Utility based on a sequence of actions
  - Model **Sequential Decision Problems**

# Example: What SEQUENCE of actions should our agent take?

- Agent can take action N, E, S, W
- Each action costs –1/25



# MDP Tuple: <S, A, P, R>

- **S:** State of the agent on the grid
  - Ex: state (4,3)

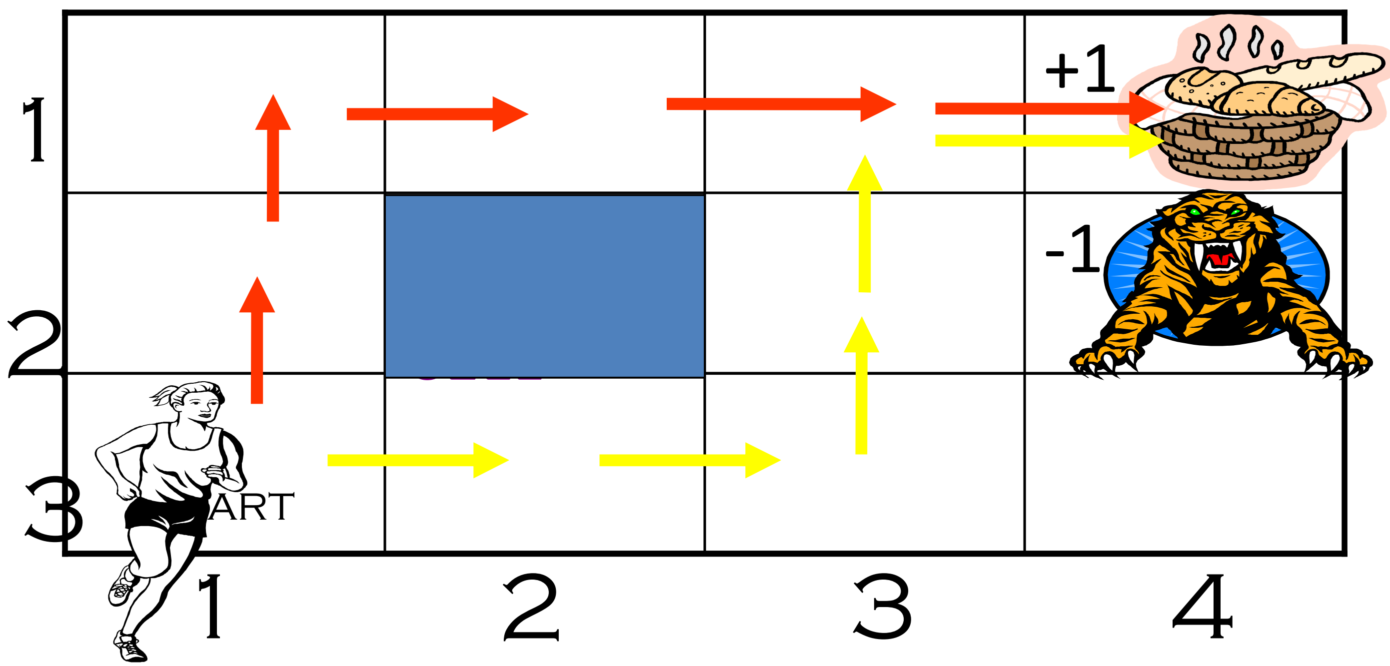
- **A:** Actions of the agent, i.e., N, E, S, W

- **P:** Transition function
  - Table P(s'| s, a), prob of s' given action "a" in state "s"
  - E.g., P( (4,3) | (3,3), N) = 0.1
  - E.g., P((3, 2) | (3,3), N) = 0.8
  - (Robot movement, uncertainty of another agent's actions,…)

- **R:** Reward
  - R( (3, 3), N) = -1/25
  - R (4,1) = +1

# How Would you Solve this Problem?

- Simple search algorithm? **Not deterministic**
- Apply MEU to an entire sequence of actions?
  - *Create multiple plans, e.g., Red plan vs. Yellow plan below*
  - *Choose a plan that leads to MEU*



# How Would you Solve this Problem?

- Apply **MEU** to an entire sequence of actions?
- Does not work because uncertainty at every step
  - E.g., After first step of Red plan, may move east not North!
  - No action specified there (I.e., in cell (2,1))

- Solution is a *Policy*
  - Complete mapping from states to actions

# MDP Basics and Terminology

- **Markov Assumption:** Transition probabilities (and rewards) from any given state depend only on the state and not on previous history

- An agent must make a decision or control a probabilistic system
  - Goal is to choose a sequence of actions for optimality
  - **Decision Epoch:** Points at which decisions are made
    - **Finite horizon MDPs**: # of decision epochs is finite i.e. fixed time after which game ends : Time dependent policy
    - **Infinite horizon MDPs:** # of decision epochs is infinite i.e. Time independent policy
  - **Transition model:** Table of probabilities P
    - In our example, 0.8, 0.1, 0.1 transition probabilities
    - P(J | S, A) : Probability of state J, given action A in State S
  - **Absorbing state:** Goal state

# Reward Function

- Reward is assumed associated with state, action i.e. **R(S, A)**
  - If all actions have the same reward can use R(S)
  - We could also assume a mix of R(S,A) and R(S)
  - Will use R(S,A) as the notation

- Sometimes, reward associated with state, action, destination-state
  - R(S,A,J)
  - $R(S,A) = \sum_{J} R(S,A,J) * P(J \mid S, A)$

# MDP Policy

- **Decision Rule:** Procedure to choose action in each state for ***a given decision epoch***
  - E.g., MDP has states, S1 and S2, with actions A1, A2 in both states
  - Decision rules Di for each decision epoch “i” as shown in table below
  - Four decision rules shown, D1, D2, D3, D4, one for each epoch
  - Numbers in (..) are probabilities, e.g., 0.7, 0.3, 1.0
- **Policy:** Decision rule to be used at all decision epochs
  - Policy = {D1, D2, D3,D4} (assuming finite horizon T = 4)

| D1 | D2 | D3 | D4… |
|---|---|---|---|
| S1→A1 (0.7)<br>→A2 (0.3)<br>S2→A2 (1.0) | S1→A1 (1.0)<br>S2→A1 (0.3)<br>→A2 (0.7) | S1→A2 (1.0)<br>S2→A2 (1.0) | …. |

# Stationary and Deterministic Policies

- **Stationary** policy implies same decision rule in every epoch
  - **Stationary policy:** {D, D, D, D…}
  - **Non-stationary policy** changes with time (e.g., D1,D2, D3…Dn)

- **Deterministic policy** implies choosing an action with certainty
  - **Deterministic policy:** Si → Ai (probability 1.0)
  - **Randomized policy:** Probability distribution on the set of actions

- What type of a policy is the following?

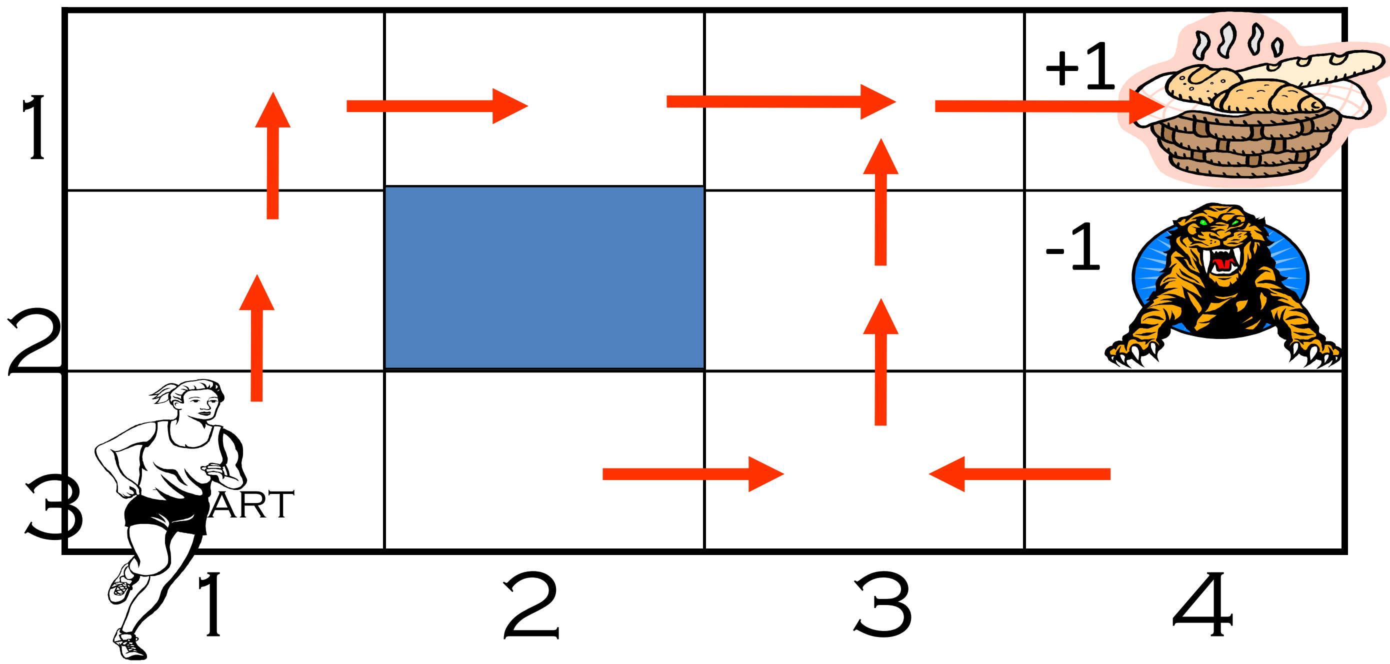
| D | D | D | D |
|---|---|---|---|
| S1→A1 (1.0)<br>S2→A2 (1.0) | S1→A1 (1.0)<br>S2→A2 (1.0) | S1→A1 (1.0)<br>S2→A2 (1.0) | S1→ A1(1.0)<br>S2→ A2(1.0) |

# Stationary and Deterministic Policies

- **Optimal** MDP policy for infinite horizon is Stationary & Deterministic policies (aka pure policy)
- Policy denoted by symbol $\pi$
- Stationary & deterministic policies denoted $\pi^{SD}$
- Is  a policy $\pi^{SR}$ possible? (SR = Stationary & randomized)

- **Note:**
  - **When nothing is specified regarding time horizon, assume infinite horizon**
  - **When asked to find the policy at time horizon = 4, it means find the decision rule D4. It can also be stated as find decision rule for T = 4 or D4.**
  - **When asked to find policy for a time horizon of 4, means find all decision rules D1, D2, D3 and D4**

# Pure Policies: $\pi^{SD}$




- Deterministic, non-changing mapping from states to actions
  - $\pi((1,3)) \rightarrow$ North (non-changing, non-random)
  - $\pi((1,2)) \rightarrow$ North
  - $\pi((4,3)) \rightarrow$ West……

# Policy

- **Policy** is like a plan, but not quite
  - Certainly, generated ahead of time, like a plan

- Unlike traditional plans, it is not a sequence of actions that an agent must execute
  - If there are failures in execution, agent can continue to execute a policy

- Prescribes an action for all the states

- Maximizes expected reward, rather than just reaching a goal state

# Value Iteration

- *Basic algorithm is very simple!*
- *Initialize: $U_0 (I) = 0$*
- Iterate:

$$U_{t+1} (I) = \max_{A} [ R(I,A) + \sum_{J} P(J|I,A)^* U_t (J) ]$$

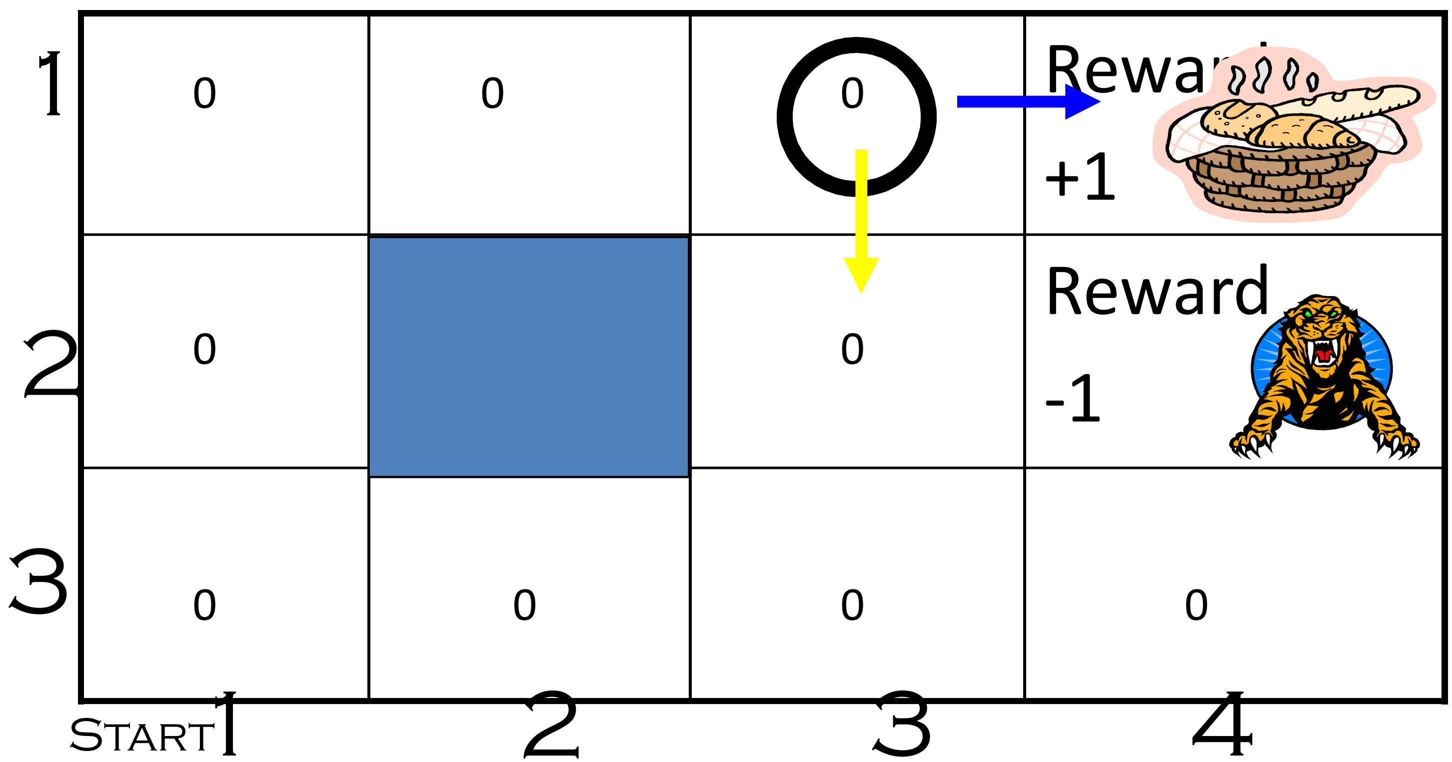
–Until close-enough ($U_{t+1}$, $U_t$)

*Dr. Richard Bellman*

- *Iteration step called "Bellman update"*
- *Inventor of dynamic programming (1957)*

# Iteration #1: Cell (3,1)

- East: -1/25 + [ 0.8 * 1 + 0.1 * 0 + 0.1 * 0 ] = 0.76
- North/South: -1/25 + [0.8 * 0 + 0.1 * 1 + 0.1 * 0 ] = 0.06
- West: ??
- So, State (3,1) has value of 0.76



# Markov Chain

# Discounting

# Value Iteration: Modify

- *Initialize:* $U_0(I) = 0$
- Iterate:

$$U_{t+1}(I) = \max_{A} \left[ R(I,A) + \gamma \sum_{J} P(J|I,A) * U_t(J) \right]$$

  – Until close-enough ($U_{t+1}$, $U_t$)

- At the end of iteration, calculate optimal policy:

$$Policy(I) = \operatorname{argmax}_{A} \left[ R(I,A) + \gamma \sum_{J} P(J|I,A) * U_{t+1}(J) \right]$$