10-601
Machine Learning

Markov decision processes (MDPs)

The weeks ahead

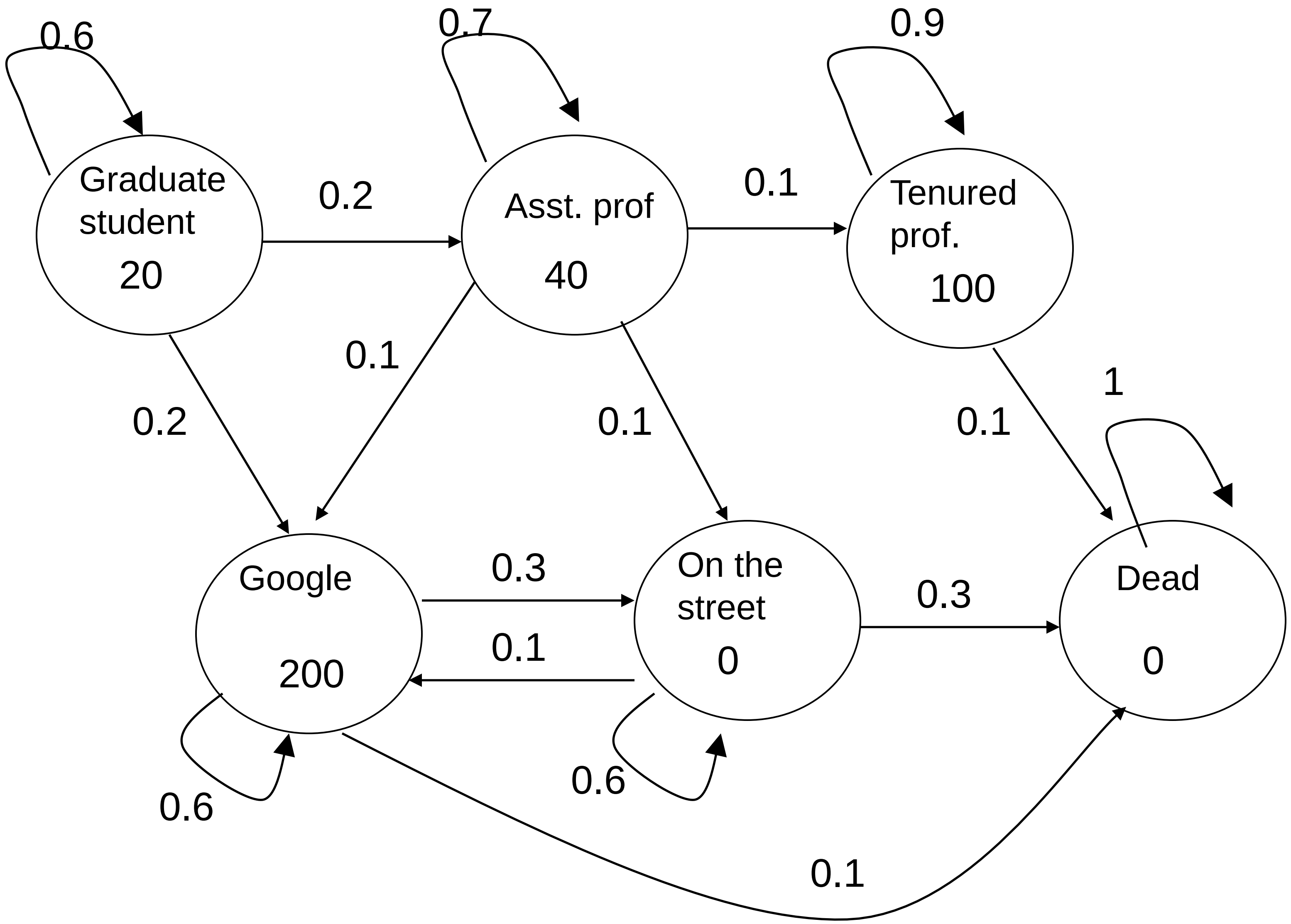
- Applications of HMM to biology
- Dimensionality reduction
- SVM
- Boosting
- Model and feature selection

Markov decision processes (MDPs)

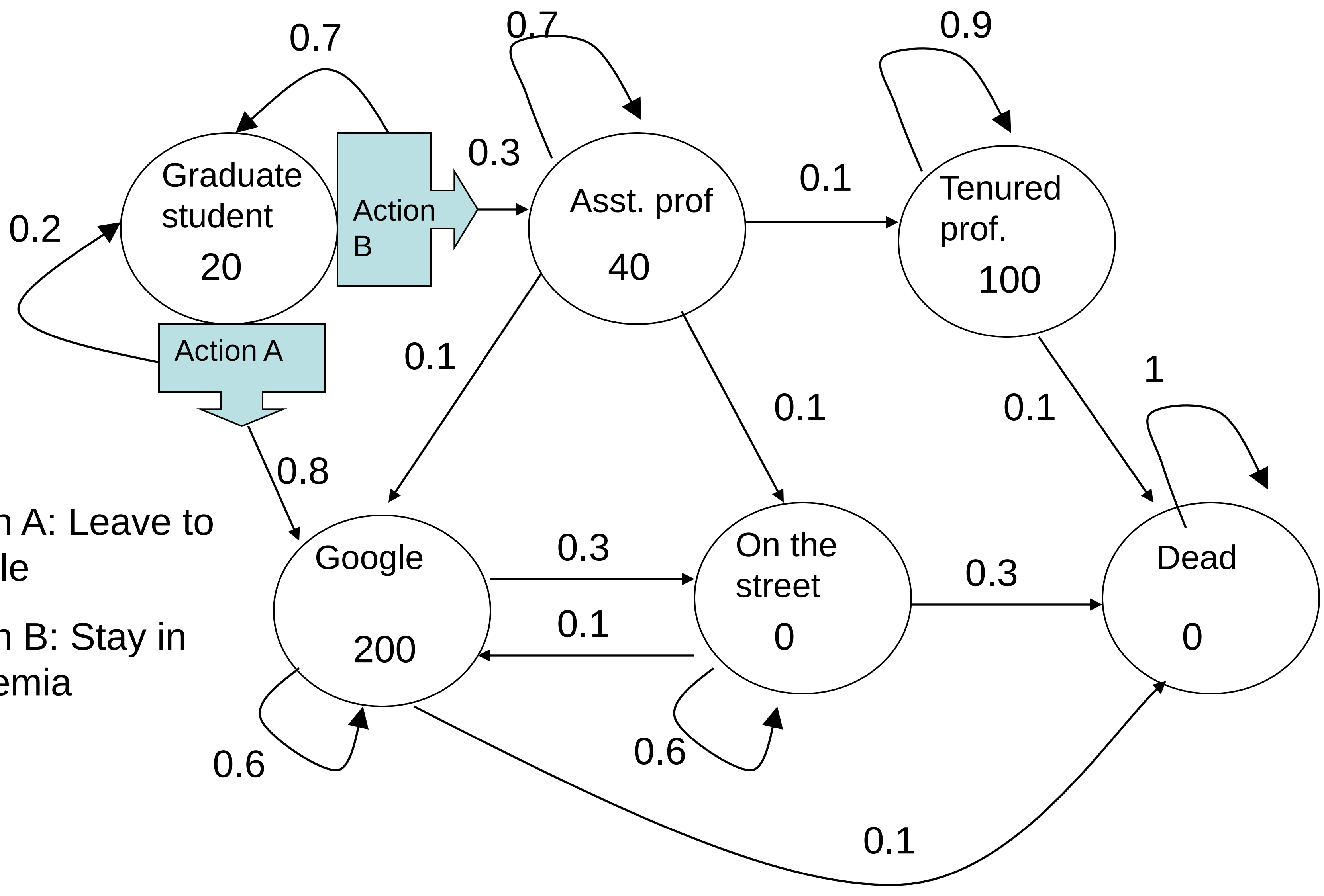
What’s missing in HMMs

- HMMs cannot model important aspects of agent interactions:
  - No model for rewards
  - No model for actions which can affect these rewards
- These are actually issues that are faced by many applications:
  - Agents negotiating deals on the web
  - A robot which interacts with its environment

Example: No actions



Example: Actions



Action A: Leave to Google

Action B: Stay in academia

Formal definition of MDPs

- A set of states $\{s_1 \ldots s_n\}$
- A set of rewards $\{r_1 \ldots r_n\}$
- A set of actions $\{a_1 .. a_m\}$
- Transition probability

One reward for each state

Number of actions could be larger than number of states

$$P_{i,j}^{k} = P(q_{t+1} = s_j \mid q_t = i \,\&\, h_t = a_k)$$

Questions

- What is my expected pay if I am in state *i*
- What is my expected pay if I am in state *i* and perform action *a*?

Solving MDPs

- No actions: Value iterations

- With actions: Value iteration, Policy iteration

Value computation

- An obvious question for such models is what is the combined expected value for each state
- What can we expect to earn over our life time if we become Asst. prof.?
- What if we go to industry?

Before we answer this question, we need to define a model for future rewards:

- The value of a current award is higher than the value of future awards

  - Inflation, confidence

  - Example: Lottery

Discounted rewards

- The discounted rewards model is specified using a parameter γ
- Total rewards = current reward +

γ (reward at time t+1) +

γ^2 (reward at time t+2) +

………….

γ^k (reward at time t+k) +

infinite sum

Discounted awards

- The discounted award model is specified using a parameter γ
- Total awards = current award +

γ (award at time t+1) +

γ^2 (award at time t+2) +

………….

γ^k (award at time t+k) +

infinite sum

Converges if $0<\gamma<1$

Determining the total rewards in a state

- Define $J^*(s_i)$ = expected discounted sum of rewards when starting at state s_i
- How do we compute $J^*(s_i)$?

Factors expected pay for all possible transitions for step *i*

$$J^*(s_i) = r_i + \gamma X$$
$$= r_i + \gamma(p_{i1}J^*(s_1) + p_{i2}J^*(s_2) + \cdots p_{in}J^*(s_n))$$

How can we solve this?

Computing j*(s_i)

$$J^*(s_1) = r_1 + \gamma(p_{11}J^*(s_1) + p_{12}J^*(s_2) + \cdots p_{1n}J^*(s_n))$$

$$J^*(s_2) = r_2 + \gamma(p_{21}J^*(s_1) + p_{22}J^*(s_2) + \cdots p_{2n}J^*(s_n))$$

$$J^*(s_n) = r_n + \gamma(p_{n1}J^*(s_1) + p_{n2}J^*(s_2) + \cdots p_{nn}J^*(s_n))$$

- We have n equations with n unknowns
- Can be solved in close form

Iterative approaches

- Solving in closed form is possible, but may be time consuming.
- It also doesn't generalize to non-linear models
- Alternatively, this problem can be solved in an iterative manner
- Lets define $J^t(s_i)$ as the expected discounted rewards after *t* steps
- How can we compute $J^t(s_i)$?

$$J^1(S_i) = r_i$$

$$J^2(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^1(s_k)\right)$$

$$J^{t+1}(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^t(s_k)\right)$$

Iterative approaches

We know how to solve this!

Lets fill the dynamic programming table

- Lets define $J^k(s_i)$ as the expected discounted awards after k steps

But wait …

This is a never ending task!

$$J^2(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^1(s_k)\right)$$

$$J^{t+1}(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^t(s_k)\right)$$

When do we stop?

$$J^1(S_i) = r_i$$

$$J^2(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^1(s_k)\right)$$

$$J^{t+1}(S_i) = r_i + \gamma\left(\sum_k p_{i,k} J^t(s_k)\right)$$

Remember, we have a converging function

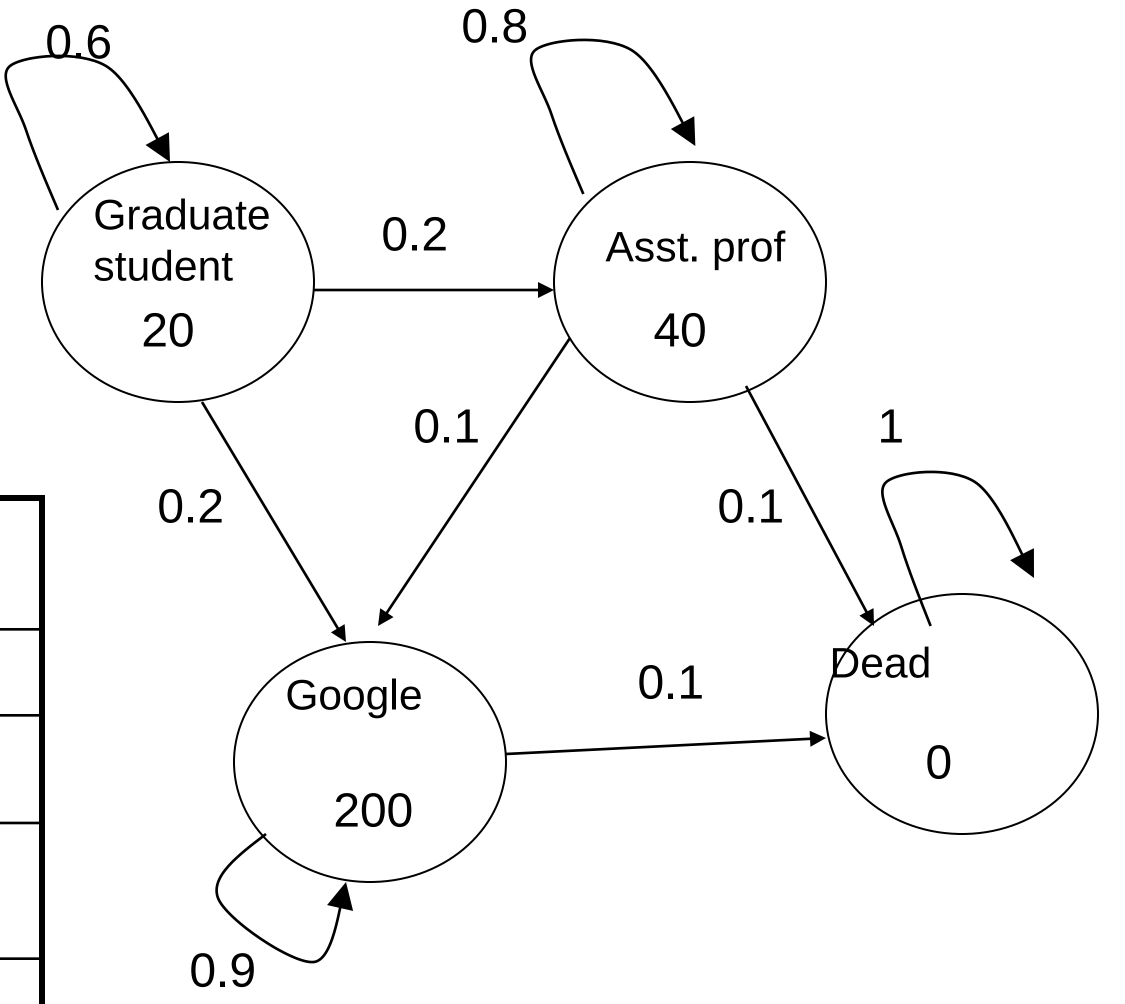
We can stop when $|J^{t-1}(s_i) - J^t(s_i)|_\infty < \varepsilon$

Infinity norm selects maximal element

Example for γ=0.9

$J^2(Gr)=20+0.9*(0.6*20+0.2*40+0.2*200)$

t	$J^t(Gr)$	$J^t(P)$	$J^t(Goo)$	$J^t(D)$
1	20	40	200	0
2	74	87	362	0
3	141	135	493	0
4	209	182	600	0



Solving MDPs

- No actions: Value iterations √

- With actions: Value iteration, Policy iteration

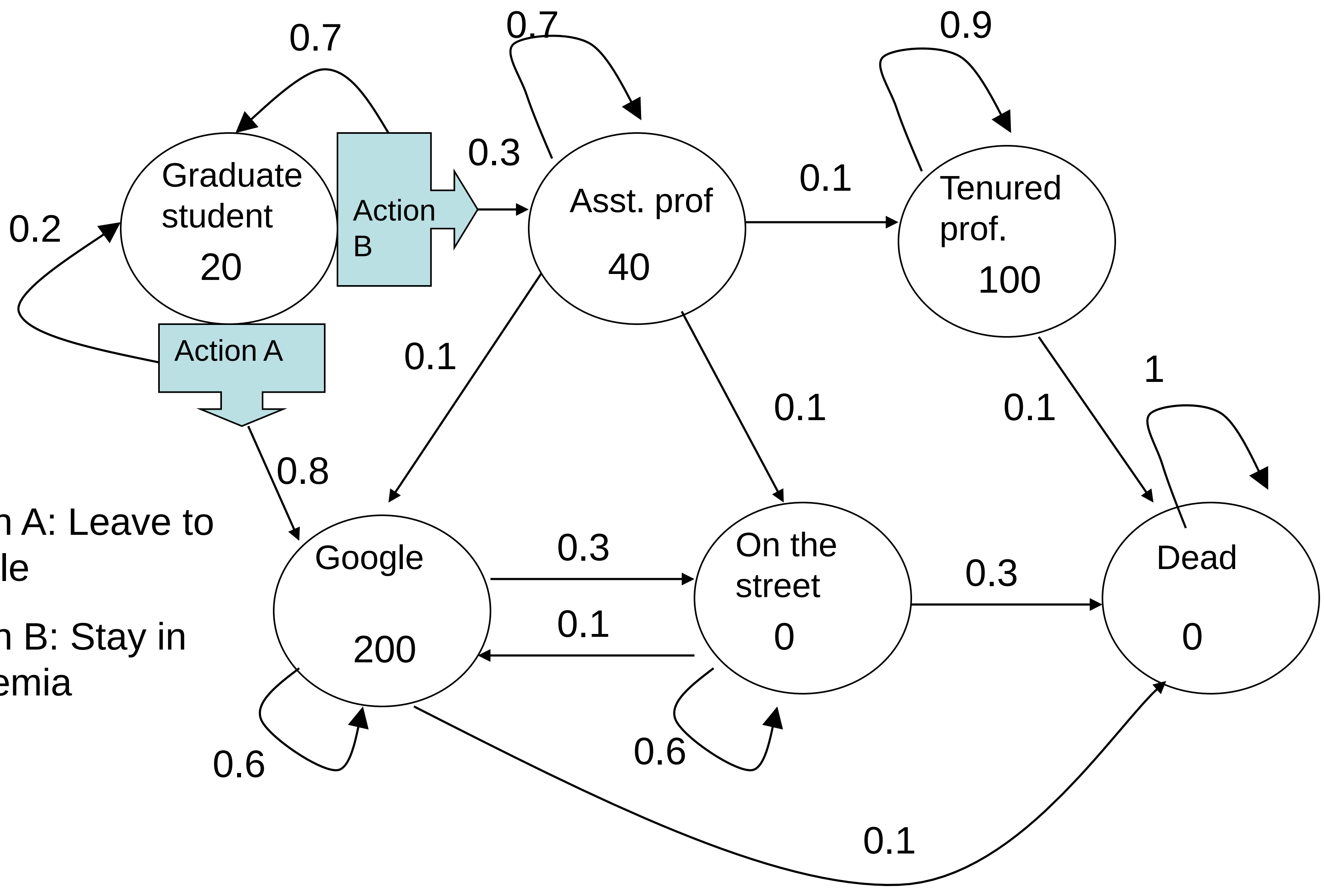
Adding actions

A Markov Decision Process:

- A set of states $\{s_1 \ldots s_n\}$
- A set of rewards $\{r_1 \ldots r_n\}$
- A set of actions $\{a_1 .. a_m\}$
- Transition probability

$$P_{i,j}^{k} = P(q_{t+1} = s_j \mid q_t = i \;\&\; h_t = a_k)$$

Example: Actions



Action A: Leave to Google

Action B: Stay in academia

Questions for MDPs

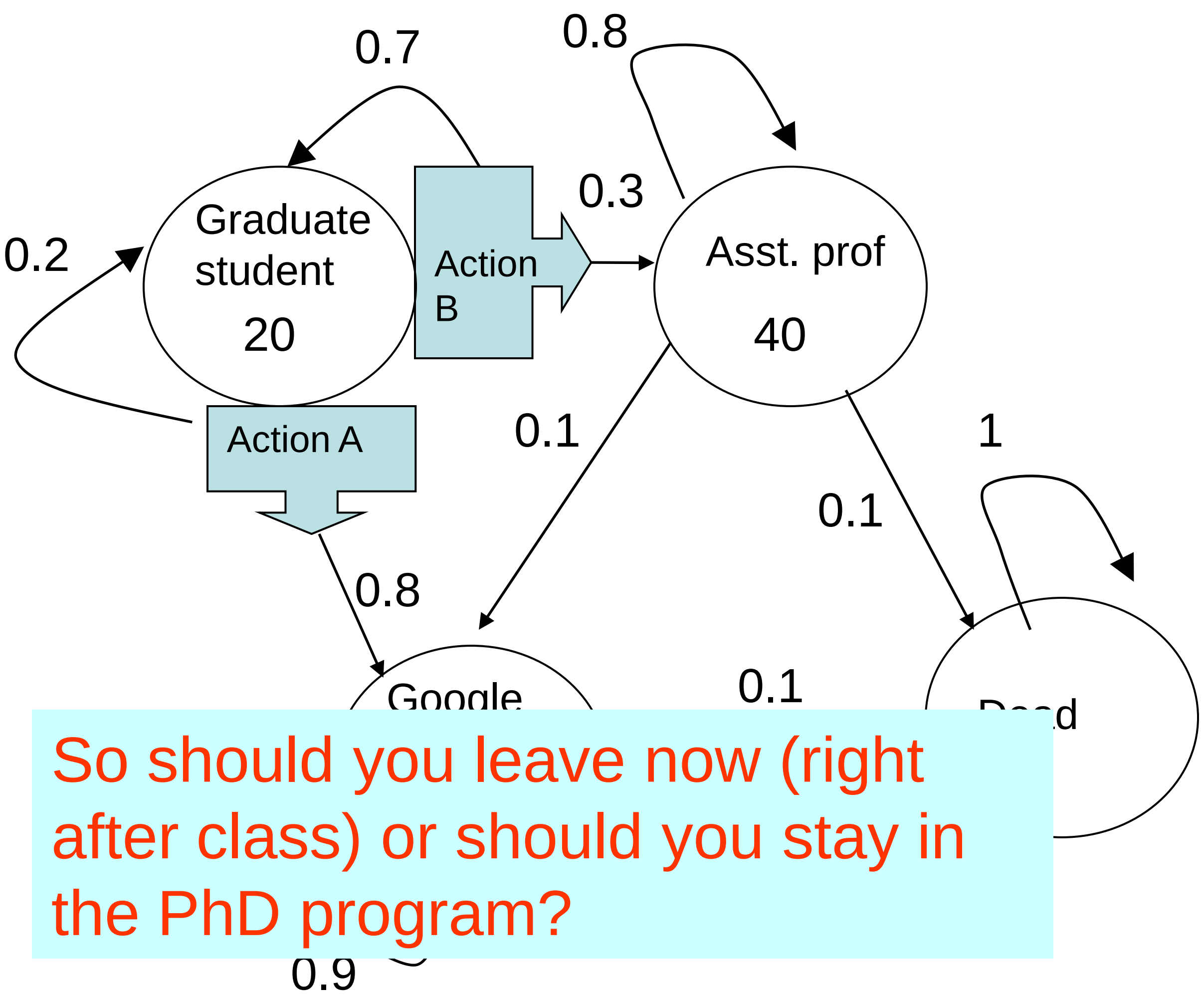
- Now we have actions
- The question changes as follows:

Given our current state and the possible actions, what is the best action for us in terms of long term payment?

Example: Actions

Action A: Leave to Google

Action B: Stay in academia



So should you leave now (right after class) or should you stay in the PhD program?

Policy

- A policy maps sates to actions
- An optimal policy leads to the highest expected returns
- Note that this does not depend on the start state

Gr	B
Go	A
Asst. Pr.	A
Ten. Pr.	B

Solving MDPs with actions

- It could be shown that for every MDP there exists an optimal policy (we won't discuss the proof).
- Such policy guarantees that there is no other action that is expected to yield a higher payoff

Computing the optimal policy: 1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define p^k_{ij} as the probability of transitioning from state i to state j when using action k
- Then we compute:

Use probabilities associated with action k

$$J^{t+1}(S_i) = \max_k r_i + \gamma\left(\sum_j p^k_{i,j} J^t(s_j)\right)$$

Also known as Bellman's equation

Computing the optimal policy: 1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define p^k_{ij} as the probability of transitioning from state i to state j when using action k
- Then we compute:

$$J^{t+1}(S_i) = \max_k r_i + \gamma \left(\sum_j p^k_{i,j} J^t(s_j) \right)$$

Run until convergences

Computing the optimal policy: 1. Modified value iteration

- We can compute it by modifying the value iteration method we discussed.
- Define p^k_{ij} as the probability of transitioning from state i to state j when using action k
- Then we compute:

$$J^{t+1}(S_i) = \max_k r_i + \gamma \left(\sum_j p^k_{i,j} J^t(s_j) \right)$$

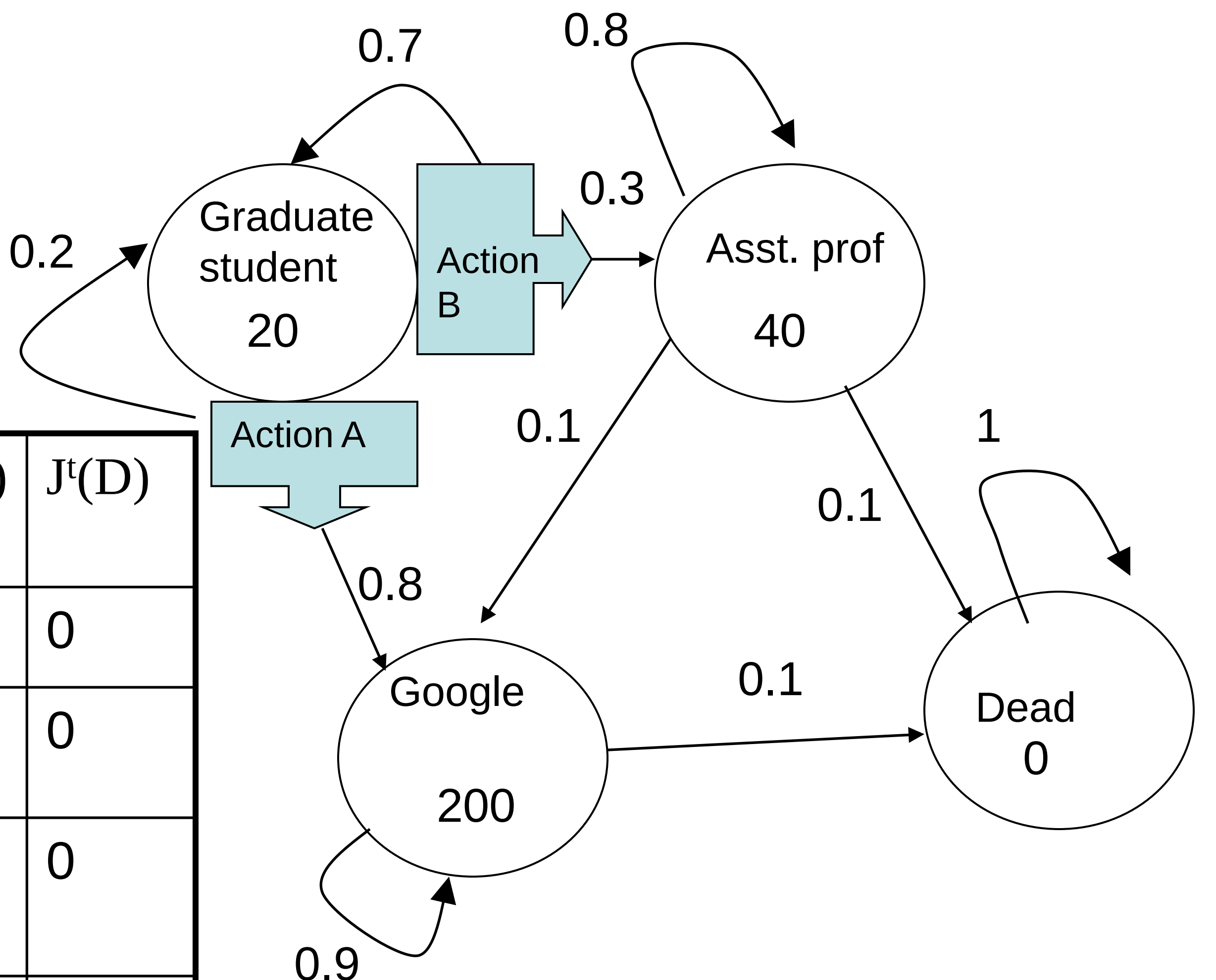
• When the algorithm converges, we have computed the best outcome for each state

• We associate states with the actions that maximize their return

Value iteration for γ=0.9

J2(Gr) = 20+ 0.9*Max {
0.2*20+0.8*200,
0.7*20+0.3*40}

t	J^t(Gr)	J^t(P)	J^t(Goo)	J^t(D)
1	20	40	200	0
2	168(A) 51(B)	87	362	0
3	311(A) 149(B)	135	493	0
4	431(A) 252(B)	182	600	0



Computing the optimal policy: 2. Policy iteration

- We can also compute optimal policies by revising an existing policy.
- We initially select a policy at random (mapping from states to actions).
- We re-compute the expected long term reward at each state using the selected policy
- We select a new policy using the expected rewrads and iterate until convergences

Policy iteration: algorithm

- Let $\pi_t(s_i)$ be the selected policy at time t

1. Randomly chose π_0 ; set t = 0
2. For each state s_i compute $J^*(s_i)$, the long term expected reward using policy π_t .
3. Set $\pi_t(s_i)$ = $\max_k r_i + \gamma \left(\sum_j p_{i,j}^k J^*(s_j) \right)$
4. Convergence? Yes: output policy. No: t = t + 1, go to 2.

Policy iteration: algorithm

- Let $\pi_t(s_i)$ be the selected policy at time t

1. Randomly chose π_0 ; set t = 0
2. For each state s_i compute $J^*(s_i)$, the long term expected reward using policy π_t .
3. Set $\pi_t(s_i)$ = $\max_k r_i + \gamma\left(\sum_j p_{i,j}^k J^*(s_j)\right)$
4. Convergence? Yes: output policy. No: t = t + 1, go to 2.

Can be computed using $J^*(s_i)$ for all states

Once the policy is fixed we are back to rewards only models, so this can be computed using value iteration

Value iteration vs. policy iteration

- Depending on the model and the information at hand:

  - If you have a good guess regarding the optimal policy then policy iteration would converge much faster

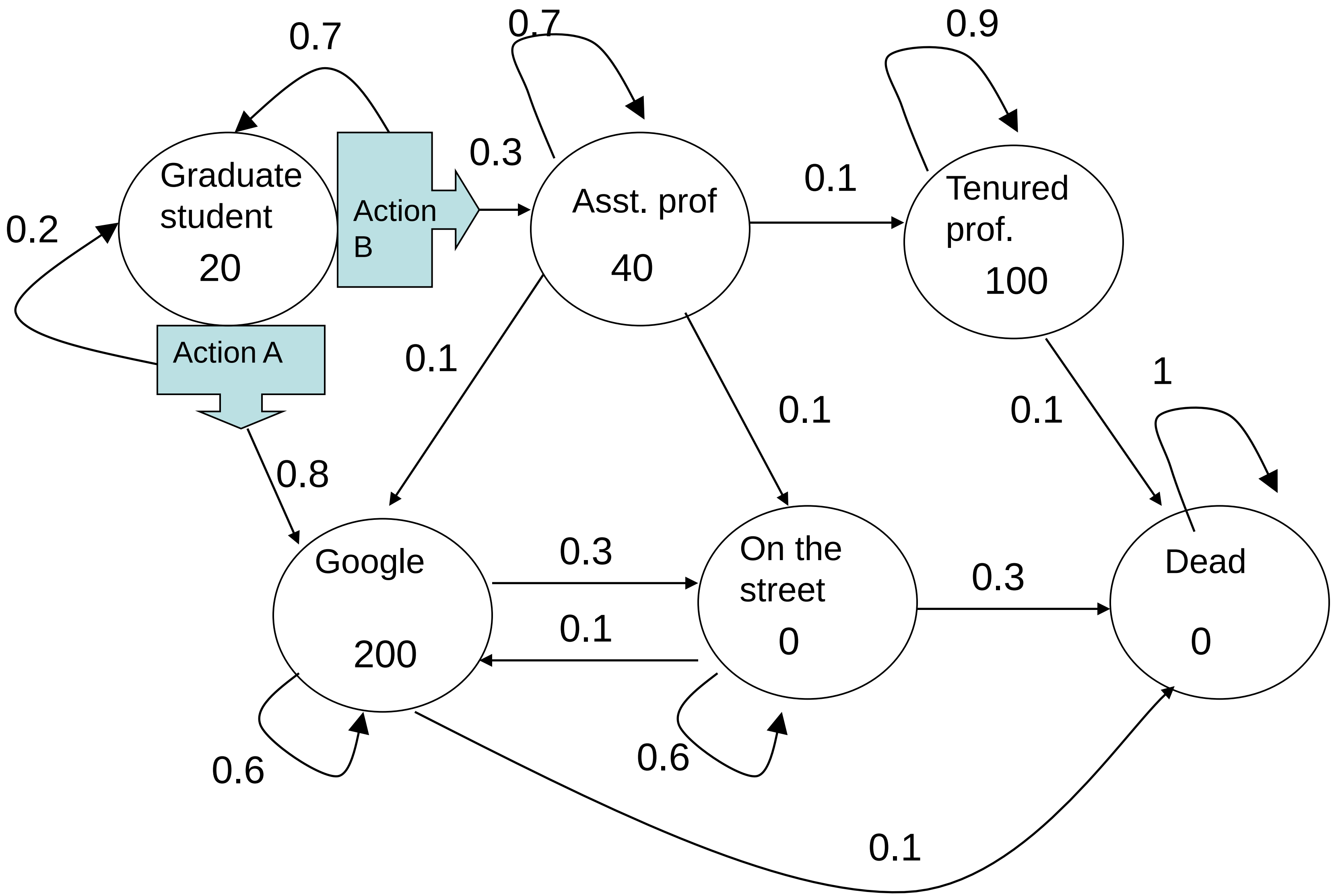
  - similarly, if there are many possible actions, policy iteration might be faster

  - otherwise value iteration is a safer way

Demo

Reinforcement learning (RL)

MDP with actions: How do we learn the model?



From MDPs to Reinforcement Learning (RL)

1. We do not assume we know the Markov model
2. We learn the model from observations (online)

- Examples:
  - Game playing
  - Robot interacting with enviroment
  - Agents

What you should know

- Models that include rewards and actions
- Value iteration for solving MDPs
- Policy iteration