Markov Decision Processes & Inverse Reinforcement Learning

Hal Daumé III

Computer Science
University of Maryland

me@hal3.name

CS 422: Introduction to ML

Many slides courtesy of Dan Klein, Stuart Russell, and Andrew Moore

Videos courtesy of Andrew Ng & Drew Bagnell

Reinforcement Learning

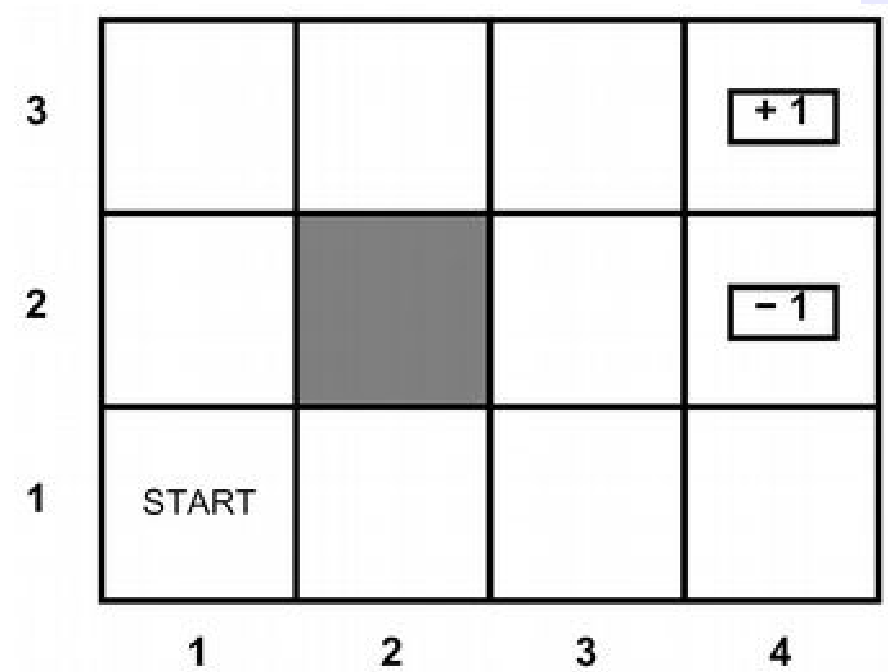
- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must learn to act so as to maximize expected rewards
  - Change the rewards, change the learned behavior
- Examples:
  - Playing a game, reward at the end for winning / losing
  - Vacuuming a house, reward for each piece of dirt picked up
  - Automated taxi, reward for each passenger delivered

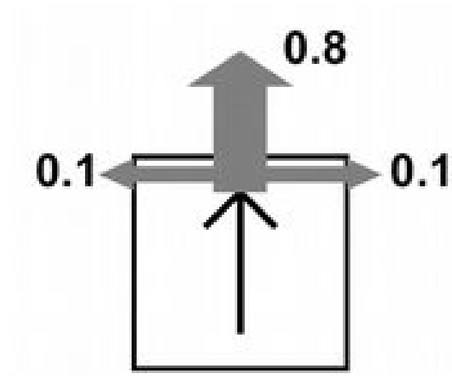
Human Reinforcement Learning

Markov Decision Processes

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s,a,s')
    - Prob that a from s leads to s'
    - i.e., P(s' | s,a)
    - Also called the model
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state (or distribution)
  - Maybe a terminal state

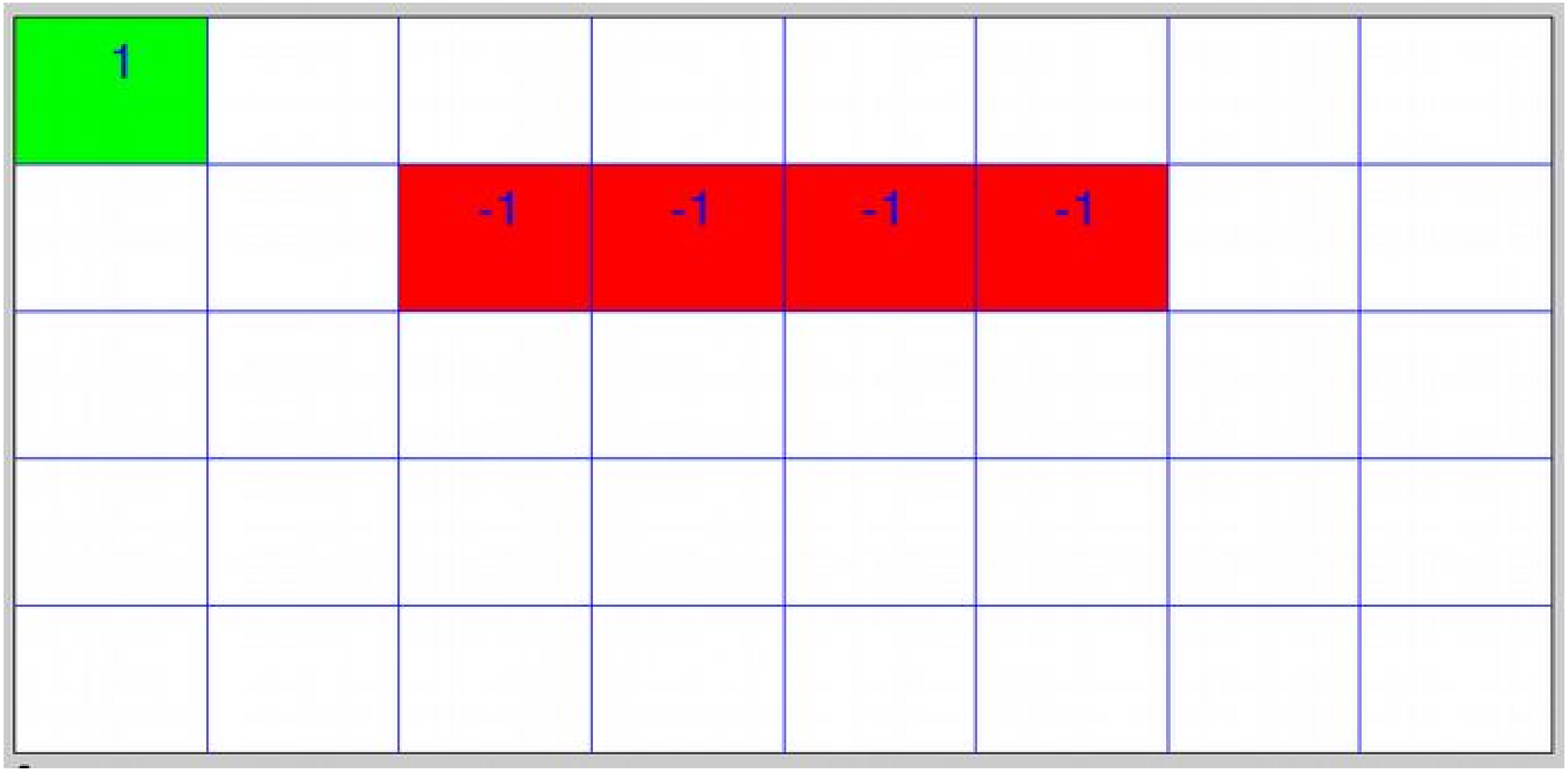
- MDPs are a family of non-deterministic search problems
  - Reinforcement learning: MDPs where we don't know the transition or reward functions





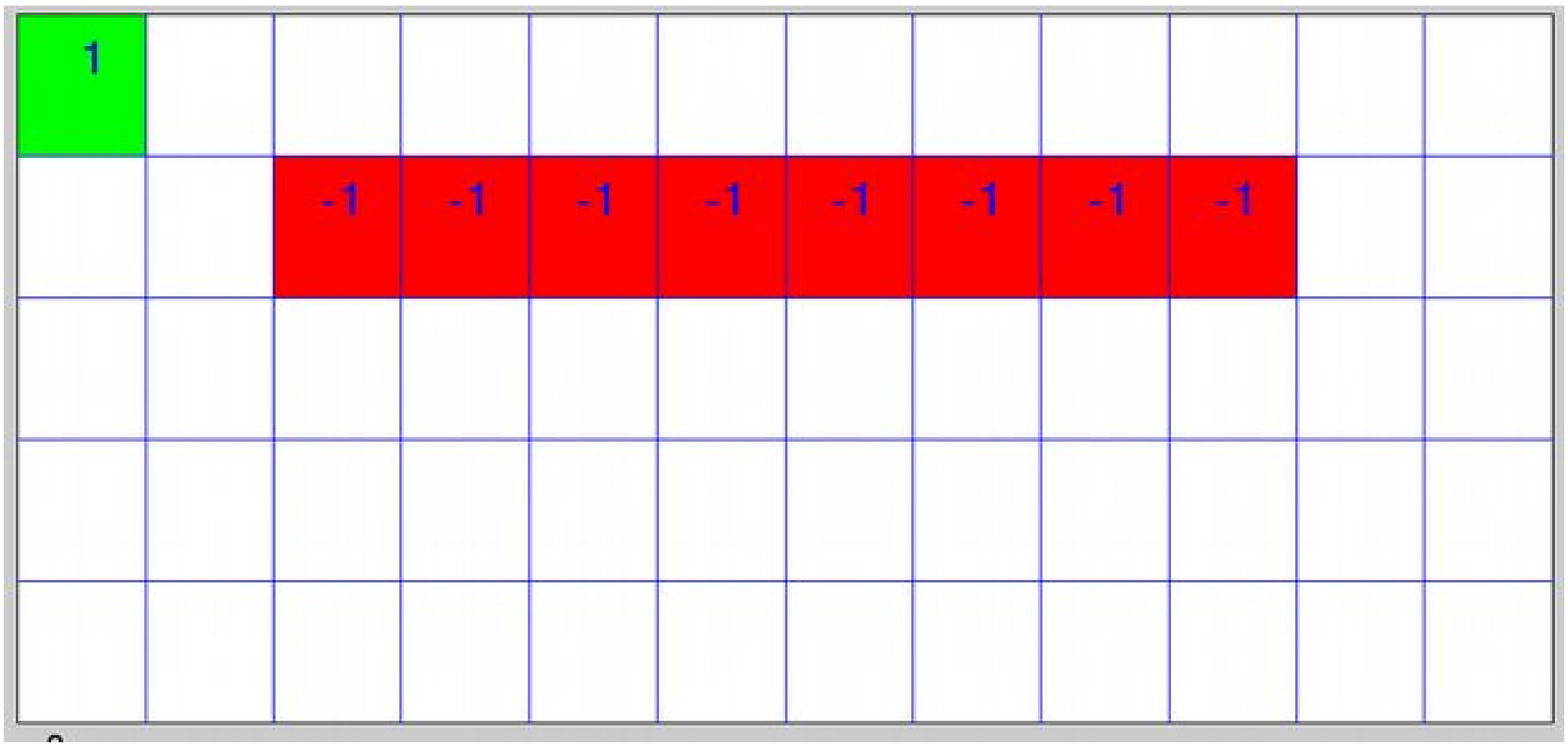
Map 0: Would you go across the top?

- Start in top-right, +\$1 for top left, -\$1 for red squares
- Costs *N* cents per step
- For what value *N* would you risk the "high road"?
  - Write something between 1 cent and 12 cents



Map 1: Would you go across the top?

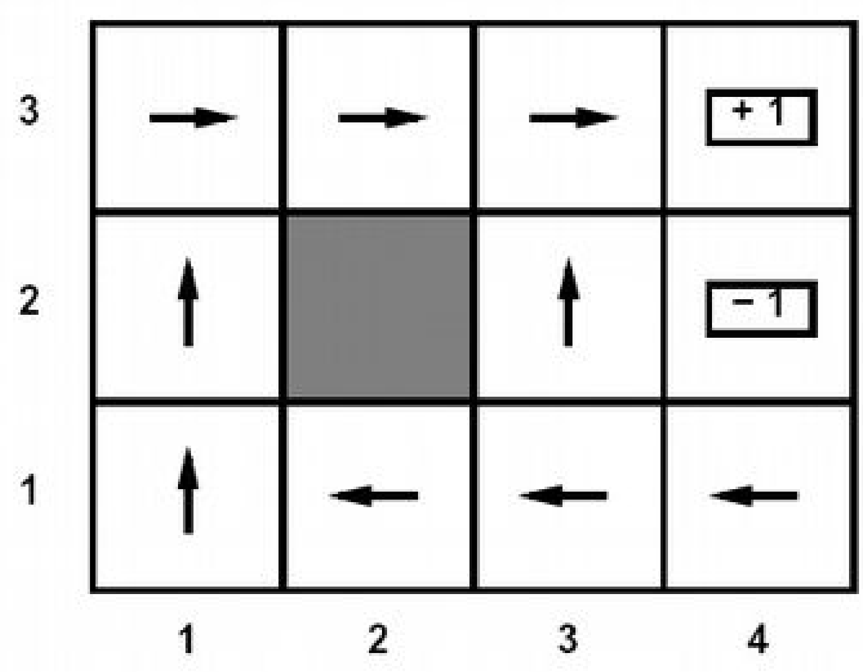
- Start in top-right, +$1 for top left, -$1 for red squares
- Costs *N* cents per step
- For what value *N* would you risk the "high road"?
  - Write something between 1 cent and 12 cents



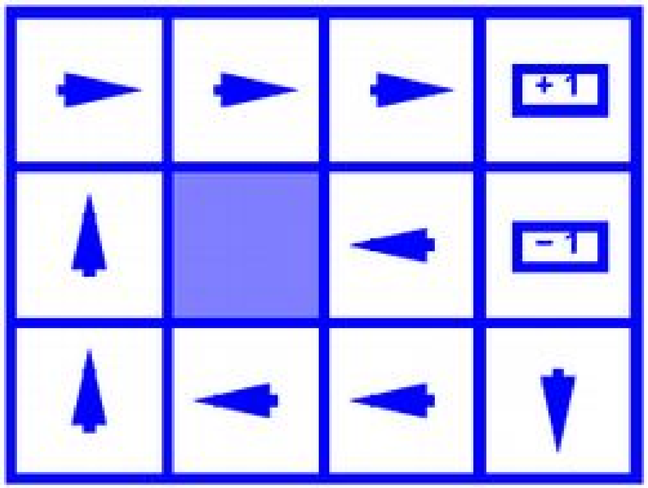
Solving MDPs

- In deterministic single-agent search problem, want an optimal plan, or sequence of actions, from start to a goal
- In an MDP, we want an optimal policy $\pi(s)$
  - A policy gives an action for each state
  - Optimal policy maximizes expected if followed
  - Defines a reflex agent

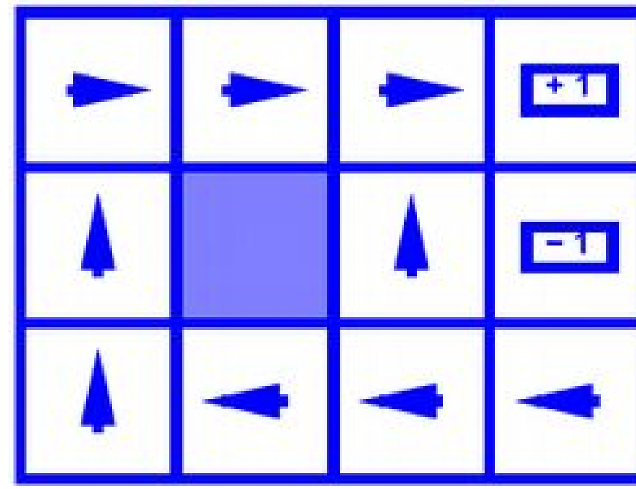
Optimal policy when R(s, a, s') = -0.04 for all non-terminals s



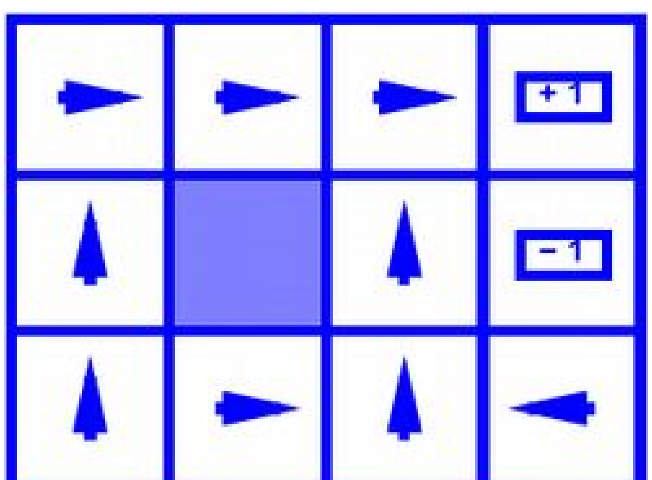
Example Optimal Policies



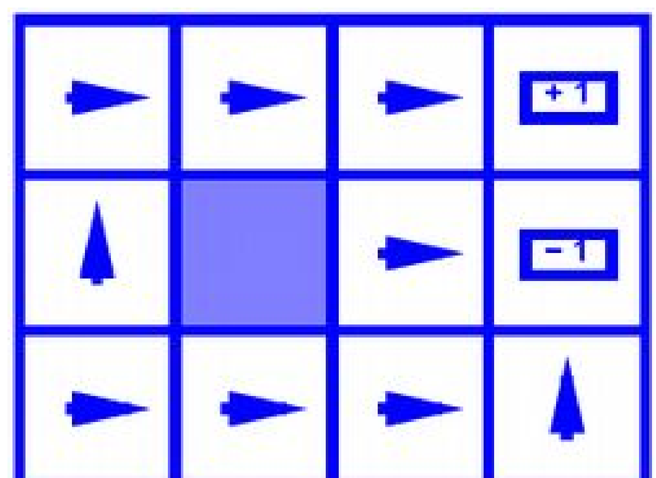
R(s) = -0.01



R(s) = -0.03



R(s) = -0.4



R(s) = -2.0

Inverse RL: Motivation

- Given: (1) measurements of an agent's behavior over time, in a variety of circumstances, (2) if needed, measurements of the sensory inputs to that agent; (3) if available, a model of the environment.
- Determine: the reward function being optimized.

Why?

- Reason #1: Computational models for animal and human learning.
- "In examining animal and human behavior we must consider the reward function as an unknown to be ascertained through empirical investigation."
- Particularly true of multiattribute reward functions (e.g. Bee foraging: amount of nectar vs. flight time vs. risk from wind/predators)

Why?

- Reason #2: Agent construction.
- “An agent designer [...] may only have a very rough idea of the reward function whose optimization would generate 'desirable' behavior.”
- e.g. “Driving well”
- Apprenticeship learning: Recovering expert's underlying reward function more “parsimonious” than learning expect's policy?

Applications in multi-agent systems

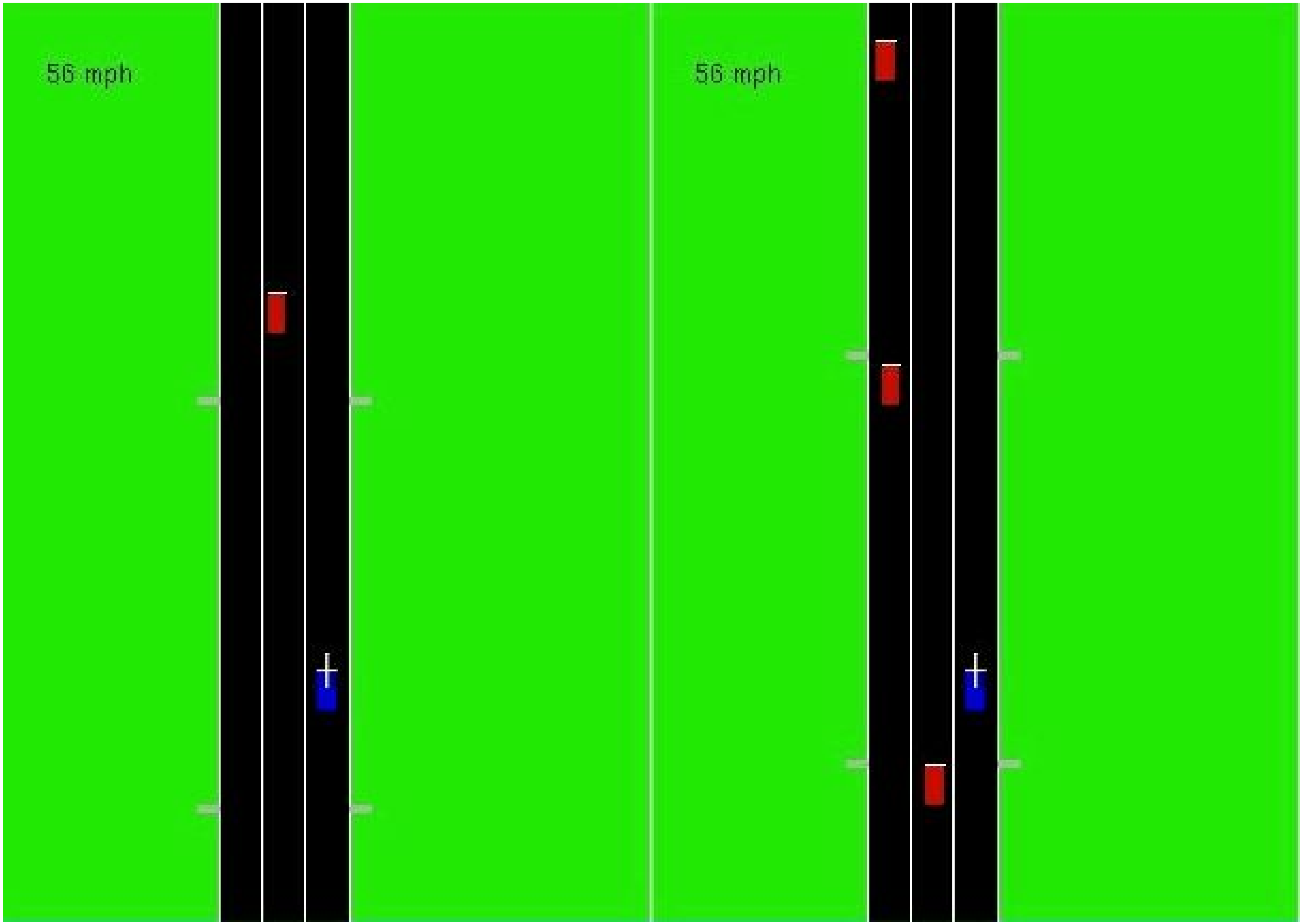
- In multi-agent adversarial games, learning opponents' reward functions that guild their actions to devise strategies against them.
- In mechanism design, learning each agent's reward function from histories to manipulate its actions.
- and more?

Car Driving Experiment

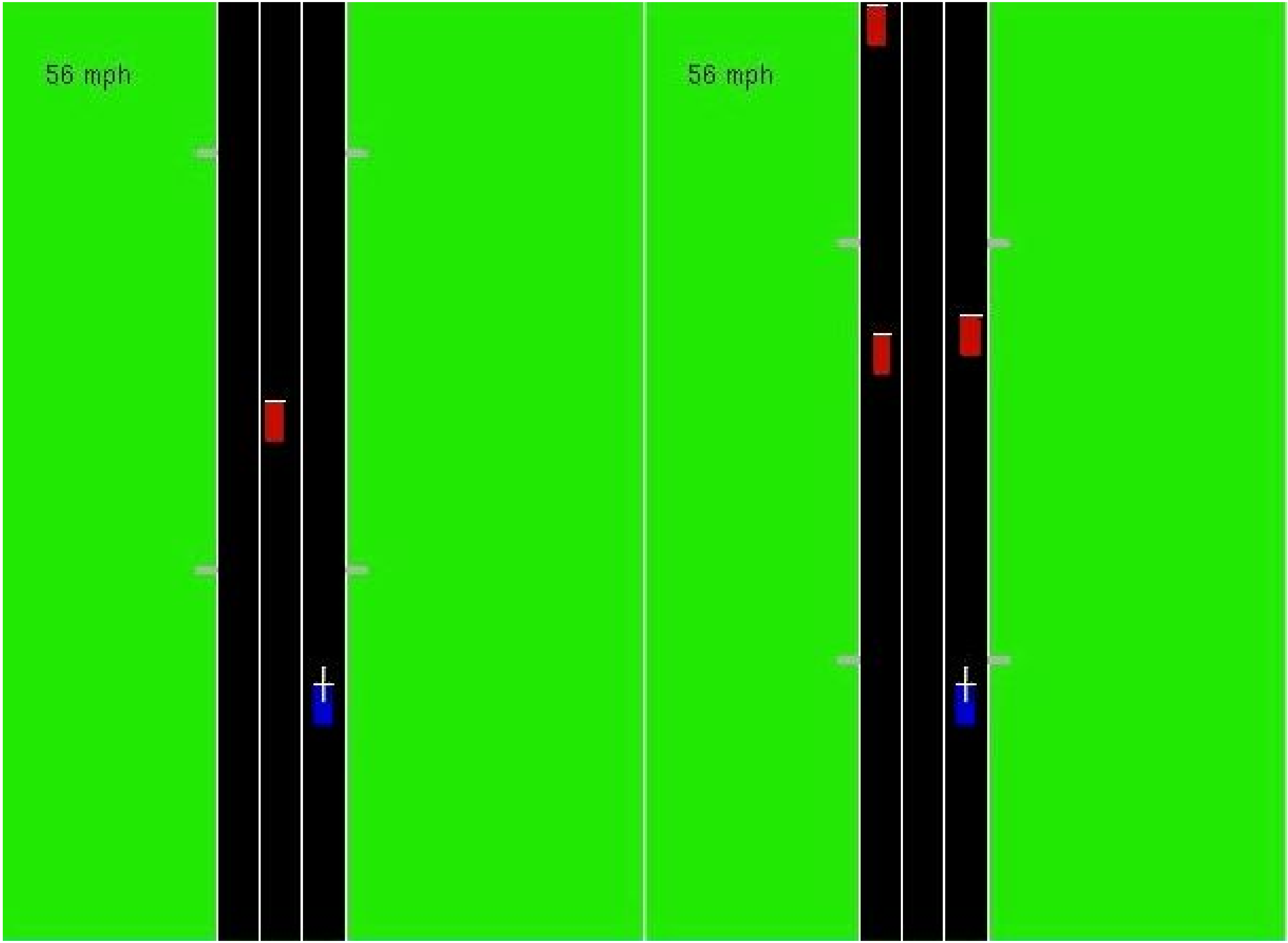
- No explicit reward function at all!
- Expert demonstrates proper policy via 2 min. of driving time on simulator (1200 data points).
- 5 different “driver types” tried.
- Features: which lane the car is in, distance to closest car in current lane.
- Algorithm run for 30 iterations, policy hand-picked.
- Movie Time! (Expert left, IRL right)

[Abbeel+Ng, ICML04]

“Nice” driver



“Evil” driver



IRL from Sample Trajecto

Warning: need to be careful to avoid trivial solutions!

- Optimal policy available through sa
(eg., driving a car)
- Want to find *Reward* function that makes this policy look *as good as possible*
- Write $R_w(s) = w\phi(s)$ so the reward is linear

and $V_w^\pi(s_0)$ be the value of the starting state

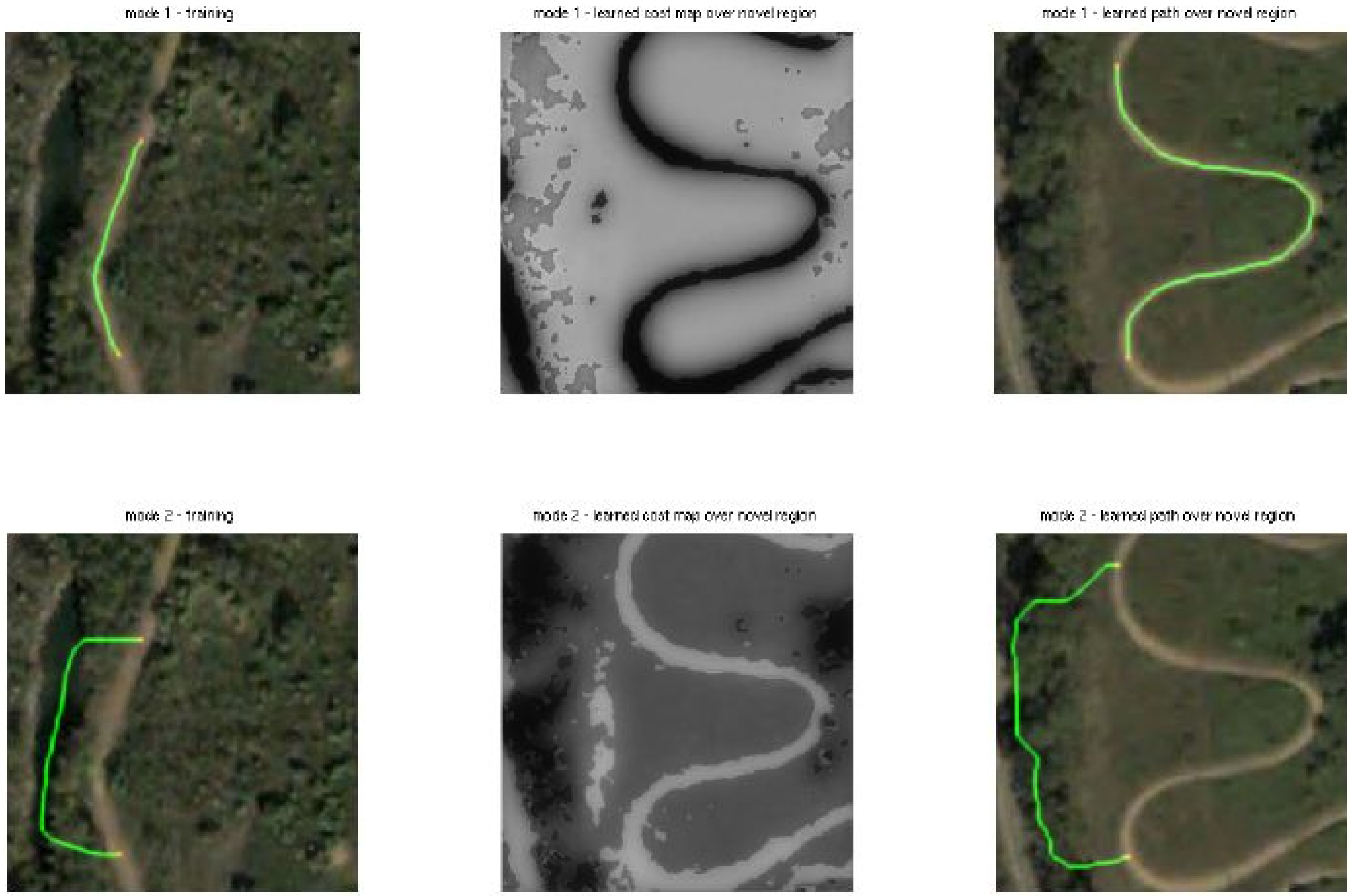
$$\max_{\mathbf{w}} \sum_{k=1}^{K} f\left(V_w^{\pi^*}(s_0) - V_w^{\pi_k}(s_0)\right)$$

How good does the "optimal policy" look?

How good does the some other policy look?

[Ng+Russell, ICML00]

Path Planning



[Ratliff+al, NIPS05]

Maximum margin planning

- Let μ(s,a) denote the probability of reaching q-state (s,a) under current model w

$$\max_{\mathbf{w}} \quad \text{margin} \quad s.t. \quad \text{planner run with w yields human output}$$

Q-state visitation frequency by human

$$\min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \quad s.t. \quad \mu(s,a)\mathbf{w}\cdot\phi(x_n,s,a) - \hat{\mu}(s,a)\mathbf{w}\cdot\phi(x_n,s,a) \geq 1, \quad \forall n,s,a$$

Q-state visitation frequency by planner

All trajectories, and all q-states

[Ratliff+al, NIPS05]

Optimizing MMP

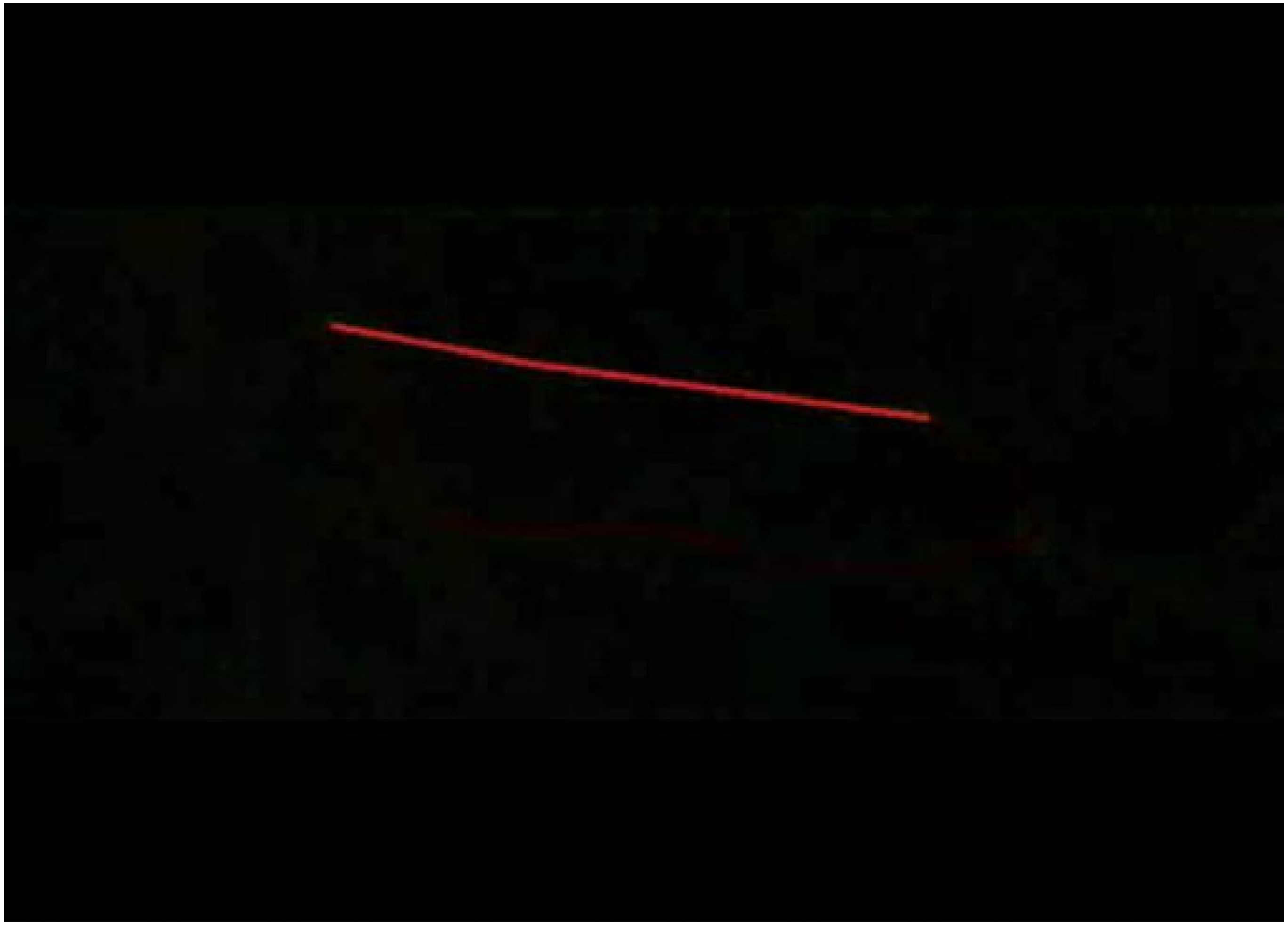
MMP Objective

SOME MATH

- For n=1..N:
  - Augmented planning:
    Run A* on current (augmented) cost map
    to get q-state visitation frequencies $\mu(s,a)$
  - Update: $\mathbf{w} = \mathbf{w} + \sum_s \sum_a [\hat{\mu}(s,a) - \mu(s,a)] \phi(x_n, s, a)$
  - Shrink: $\mathbf{w} = \left(1 - \frac{1}{CN}\right) \mathbf{w}$

[Ratliff+al, NIPS05]

Maximum margin planning movies



[Ratliff+al, NIPS05]

Maximum margin planning movies