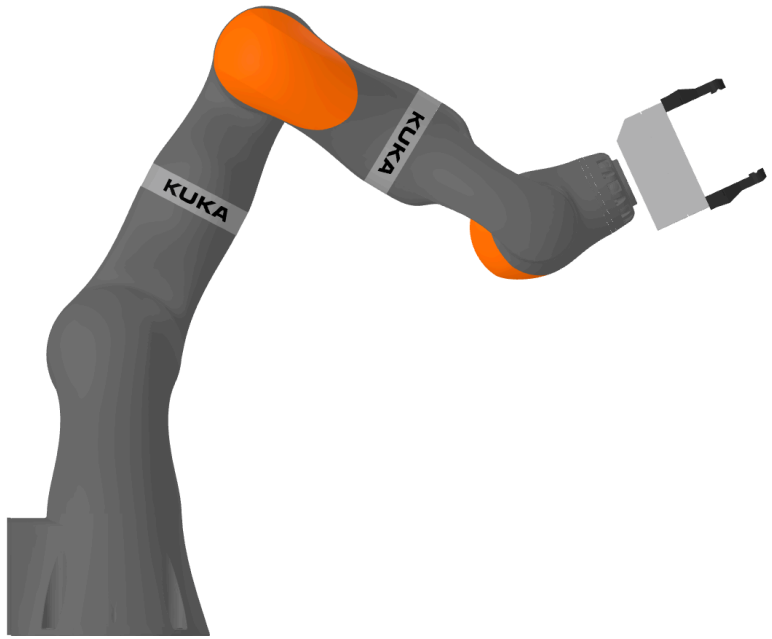


Model-based Learning I

MIT 6.4210/2

Robotic Manipulation

Fall 2026, Lecture 12

Massachusetts Institute of Technology

Reinforcement learning

Lecture 12: learning rewards and models

Leslie Kaelbling

6.4210 Fall 2026

Learning a model

Policies answer “what should I do?”
Models answer “what would happen if I did this?”

- Predicted next states let us compare possible futures before acting
- Changing the reward requires re-learning a policy or value function
- A transition model can be reused when the task changes but the dynamics do not

Goal-conditioned reinforcement learning

The goal becomes another input: $\pi(a \mid s, g)$ and $Q(s, a, g)$

$$R(s, a, g) = \begin{cases} 0, & s \in \mathcal{G}(g) \\ -C(s, a), & s \notin \mathcal{G}(g) \end{cases}$$

- In a stochastic shortest-path problem, we use $\gamma = 1$
- Nonterminal costs satisfy $C(s, a) \geqslant c_{\min} > 0$, and some policy has finite expected total cost
- We pay for the action that enters the goal set, then the episode ends

Maximizing return minimizes the expected cost of reaching the goal

g: goal description $\quad \mathcal{G}(g)$: states satisfying it $\quad C(s, a)$: step cost

Reusing experience

If the physical dynamics stay the same, an off-policy learner can relabel

$$((s, g_1), a, r, (s', g_1)) \quad \longrightarrow \quad ((s, g_2), a, R(s, a, g_2), (s', g_2))$$

- Recompute both the reward and whether s' is terminal for g_2
- Omit transitions starting at an already-satisfied goal
- *Hindsight experience replay* relabels with goals actually reached

Goal-reaching costs as quasimetrics

For exact state goals, we use the same dynamics and positive step costs, with undiscounted return

$$d(s, g) = -V^*(s, g), \qquad d(s, g) \leqslant d(s, u) + d(u, g)$$

- Nonnegative, zero exactly when $s = g$, and obeys the triangle inequality
- Need not be symmetric: dropping an object is easier than retrieving it
- Quasimetric learning builds this structure into a learned representation

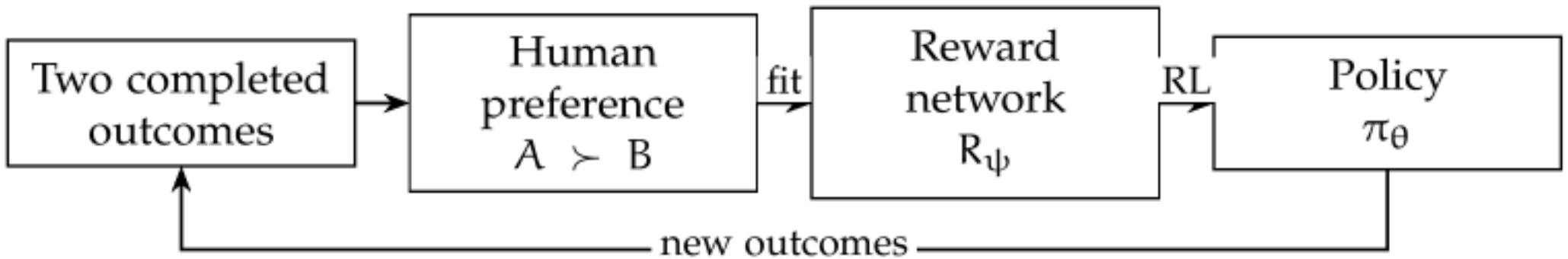
If no policy reaches the goal with finite expected cost, $d(s, g) = \infty$

$d(s, g)$: minimum expected goal-reaching cost u: intermediate state
Wang et al., Optimal Goal-Reaching RL via Quasimetric Learning

Learning a reward model

"Set the table nicely" may be easier to judge than to write as a reward

- People can label success, give scores, or compare completed outcomes
- Supervised learning fits a reward network; RL then uses its predictions



R_ψ: reward network π_θ: policy
Christiano et al., Deep Reinforcement Learning from Human Preferences

Comparing two final states

For two outcomes of the same task, the network predicts terminal rewards

$$r_A = R_\psi(s_A), \qquad r_B = R_\psi(s_B)$$

The *Bradley–Terry model* predicts the person's preference:

$$p = p_\psi(A \succ B) = \frac{e^{r_A}}{e^{r_A} + e^{r_B}} = \frac{1}{1 + e^{-(r_A - r_B)}}$$

- Only the score difference matters; equal scores give probability 1/2
- The training example tells us which outcome the person prefers

s_A, s_B: final states $A \succ B$: person prefers A

A reward update from one preference

Suppose the person prefers A, and write $\Delta = r_A - r_B$

$$\mathcal{L}(\psi) = -\log p = \log(1 + e^{-\Delta}), \qquad \frac{\partial \mathcal{L}}{\partial \Delta} = p - 1$$

$$\nabla_\psi \mathcal{L} = (p-1)\left[\nabla_\psi R_\psi(s_A) - \nabla_\psi R_\psi(s_B)\right]$$

$$\psi \leftarrow \psi + \alpha(1-p)\left[\nabla_\psi R_\psi(s_A) - \nabla_\psi R_\psi(s_B)\right]$$

Backpropagation through the same network supplies both score gradients
The update is larger when the preference was poorly predicted

$\alpha > 0$: learning rate

Inverse reinforcement learning as a choice model

We infer a reward from the trajectories a person chooses

$$g_\psi(\tau) = \sum_{t=0}^{H-1} R_\psi(s_t, a_t), \qquad p_\psi(\tau \mid \mathcal{C}) = \frac{\exp[g_\psi(\tau)]}{\sum_{\tau' \in \mathcal{C}} \exp[g_\psi(\tau')]}$$

- A comparison has two alternatives: $\mathcal{C} = \{\tau_A, \tau_B\}$
  The resulting preference model is Bradley–Terry
- A demonstration selects from all feasible trajectories: $\mathcal{C} = \mathcal{F}$
- Higher-return choices are more likely, but need not be optimal

R_ψ: learned step reward g_ψ: total trajectory reward $\mathcal{C}$: choice set
Jeon, Milli, and Dragan, Reward-Rational (Implicit) Choice

Fitting the reward to demonstrations

The mean negative log likelihood uses the partition function Z

$$Z(\psi) = \sum_{\tau \in \mathcal{T}} e^{g_\psi(\tau)}, \qquad \mathcal{L}(\psi) = -\frac{1}{N} \sum_{i=1}^{N} g_\psi(\tau^{(i)}) + \log Z(\psi)$$

For a linear reward, trajectory features add up:

$$R_\psi(s, a) = \psi^\top \phi(s, a), \qquad \Phi(\tau) = \sum_{t=0}^{H-1} \phi(s_t, a_t)$$

$$\psi \leftarrow \psi + \alpha\big(\bar{\Phi}_D - \mathbb{E}_{p_\psi}[\Phi(\tau)]\big)$$

We increase weights for features whose counts the model underpredicts
Different rewards can explain the same demonstrations

$\tau^{(i)}$: demonstration i $\alpha > 0$: learning rate
$\bar{\Phi}_D = \frac{1}{N} \sum_i \Phi(\tau^{(i)})$: mean demonstrated feature counts

Partition functions and action probabilities

Known deterministic dynamics, fixed start and horizon H, finite actions
$Z_t(s)$ sums exponentiated rewards over all continuations from (s, t)

$$Z_H(s) = 1$$

$$Z_t(s, a) = \exp[R_\psi(s, a)]\, Z_{t+1}(f(s, a))$$

$$Z_t(s) = \sum_a Z_t(s, a), \qquad Z(\psi) = Z_0(s_{\text{init}})$$

$$\pi_{\psi,t}(a \mid s) = \frac{Z_t(s, a)}{Z_t(s)}$$

Multiplying these probabilities along a path gives $e^{g_\psi(\tau)}/Z(\psi)$
Maximum causal entropy IRL handles stochastic dynamics

$f(s, a)$: next state $Z_t(s, a)$: weight of continuations starting with a
Ziebart et al., Maximum Entropy Inverse Reinforcement Learning

Expected feature counts

After the backward pass for Z, we propagate state probabilities forward

$$D_0(s) = \mathbf{1}\{s = s_{\text{init}}\}$$

$$D_{t+1}(s') = \sum_{\substack{s,a: \\ f(s,a)=s'}} D_t(s)\,\pi_{\psi,t}(a \mid s)$$

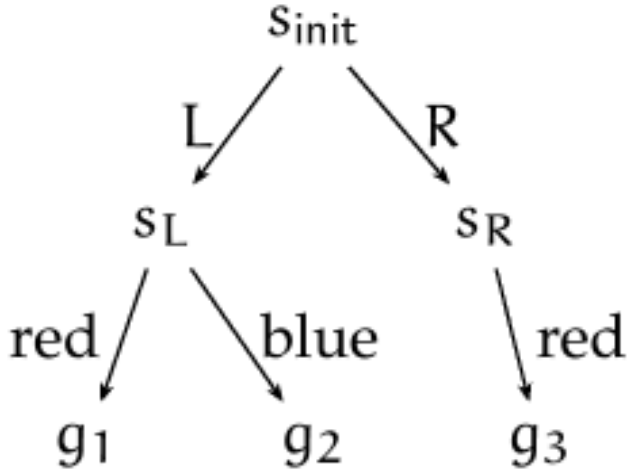
The probability of taking action a in state s at time t is $D_t(s)\pi_{\psi,t}(a \mid s)$, so

$$\mathbb{E}_{p_\psi}[\Phi(\tau)] = \sum_{t=0}^{H-1} \sum_{s,a} D_t(s)\,\pi_{\psi,t}(a \mid s)\,\phi(s,a)$$

We can compute the reward update without listing every complete trajectory

$D_t(s)$: probability of state s at time t under the current reward model

IRL example: backward weights, then action probabilities



Two steps; first reward 0
Red reward $\log 2$
Blue reward 0

Each terminal has weight $Z_2(g_j) = 1$

$$Z_1(s_L) = 2 + 1 = 3$$
$$Z_1(s_R) = 2$$
$$Z_0(s_{init}) = 3 + 2 = 5$$

$$\pi_0(L \mid s_{init}) = 3/5, \quad \pi_0(R \mid s_{init}) = 2/5$$
$$\pi_1(\text{red} \mid s_L) = 2/3, \quad \pi_1(\text{blue} \mid s_L) = 1/3$$
$$\pi_1(\text{red} \mid s_R) = 1$$

With red and blue gate counts as features, $\psi = (\log 2, 0)^T$
The first actions have equal rewards, but left offers two continuations

IRL example: expected counts and one reward update

The path probabilities follow from multiplying the action probabilities

τ	(L, red)	(L, blue)	(R, red)
$p_\psi(\tau)$	$(3/5)(2/3) = 2/5$	$(3/5)(1/3) = 1/5$	$(2/5)(1) = 2/5$

Each trajectory contains one gate, so $\mathbb{E}[\Phi] = (4/5, 1/5)^\top$
Five demonstrations contain two red gates and three blue gates
We use learning rate $1/2$

$$\psi_{\text{new}} = \begin{bmatrix} \log 2 \\ 0 \end{bmatrix} + \frac{1}{2}\left(\begin{bmatrix} 2/5 \\ 3/5 \end{bmatrix} - \begin{bmatrix} 4/5 \\ 1/5 \end{bmatrix}\right) = \begin{bmatrix} \log 2 - 1/5 \\ 1/5 \end{bmatrix}$$

We decrease the red reward and increase the blue reward
The predicted blue-path probability rises from 0.20 to about 0.27

Φ: trajectory feature counts, ordered (red, blue)

Learning transition models

Recorded transitions $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i=1}^N$ supply supervised examples

$$\mathcal{L}_{\text{reg}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|s'_i - f_\theta(s_i, a_i)\|^2$$

Squared loss fits a conditional mean, which may not be a likely outcome

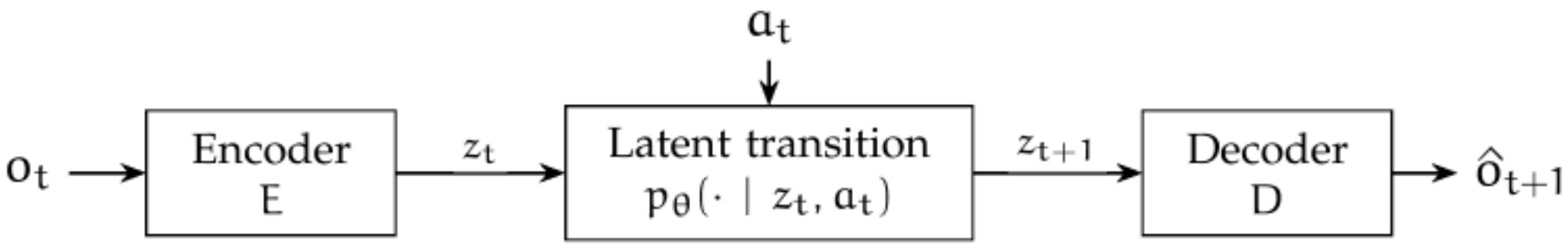
$$\mathcal{L}_{\text{NLL}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p_\theta(s'_i \mid s_i, a_i)$$

If our model can represent the true distribution, that distribution minimizes expected negative log likelihood
We can use categorical, Gaussian, diffusion, or flow models

f_θ: deterministic prediction p_θ: learned transition distribution

Latent world models

An encoder compresses observations; dynamics predict in the latent space



- Reconstruction and prediction losses train representation and dynamics
- Latent states must support prediction of observations and rewards
- Partial observations may require an encoder with memory

Reconstructing an image well does not guarantee a useful state for control

o_t: observation z_t: latent state E: encoder D: decoder

Video plans and inverse dynamics

A video model proposes a visual plan

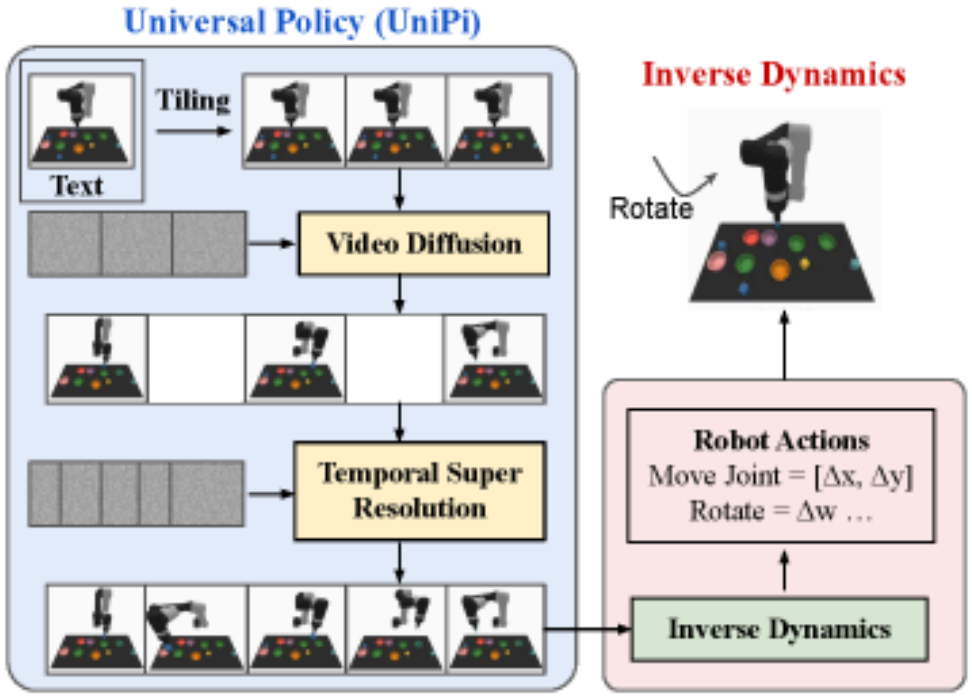
$$o_{1:H} \sim p_\theta(\cdot \mid o_0, g)$$

Inverse dynamics predicts actions

$$a_t \sim q_\psi(\cdot \mid o_t, o_{t+1})$$

It needs recorded (o_t, a_t, o_{t+1}) triples

A plausible-looking video need not be an executable plan



g: goal description o_0: current image
Du et al., UniPi, Fig. 2, cropped from the paper, CC BY 4.0