CS 690S - AI Alignment (Public)

Day: Fri, May 8 12026

Presentation Day: 2

1:
Not really sure exactly what the exact tasks they were training on were. Something related to physical intelligence thing.

2:
Multi-objective problem inference from experimentation
This one seems futile because policies that are ideal for an extremum of a given problem (feature?) don't necessary compose / combine together to create an idea policy on the pareto-front where the user is specifically.

3:

Day: Wed, May 6 12026

Presentation day!

1:
Robustness of LLM Preference alignment to Noisy Human Feedback. Ritika Hooda, Kamalesh Kumar

TLDR: Measure model learning effectiveness as noise changes over time.

cDPO and rDPO try to improve over Vanilla DPO but require knowing noise ahead of time

ROPO tries to adjust noise estimation hyperparameter based on gradient magnitude.

- How does this compare to GRPO?

2:
2D to high-fidelity 3D simulation generalization testing

3:
Diffusion behavior cloning something idk

4:
Shreya Sharma

RL on tool use to make it better, minimize tool use -> why not just reward less tool use? Or measure tool resource usage and have it minimize that?

5:
GridWorld distributional shift at test time.

Train model on R1, R2, can we get it to combine skills in practice?

Prediction: probably depends on priors (does it have a prior to combine skills?)

- This is a terrible prediction, it is always true. Need more info on the model and training How large were the models?

6:
Overfit reward functions to specific hyperparameters

- Used LLMs to generate reward functions for gridworlds

7:
SFT warm start Qwen2.5 coder-7b

8:
Cal-Sim DPO

"Cal-DPO focuses on calibrating implicit rewards against ground-truth rewards (usually one line of code) to improve performance, while SimPO focuses on efficiency by eliminating the reference model and using average sequence log-probabilities"

Can these be combined?

Day: Wed, Apr 29 12026

Today: Deception and frontiers of misalignment
Can't train out backdoors with adversarial training (why not?) what about maxent?

Emergent misalignment

Day: Fri, Apr 17 12026

Red Teaming lecture - pretty interesting.

Q: Does "Superhuman Intelligence" necessarily mean goal preservation?

- Can we study goal preservation specifically? Like what level of computation, optimization

Day: Wed, Apr 15 12026

AI Risks

People train "civilization modeling" foundation models

- They all try to predict what would be the best action, do rollouts, and find the best outcome
    - Assume these things include ideal coordination proposals, and tend to converge on them
    - People get together, fund the proposal, market satisfies demand, system continually updates as capabilities improve.

Day: Fri, Apr 10 12026

Today: Mechanistic Interpretability!
Presenter: Sankaran Vaidyanthan https://sankaranv.com/

Common Assumption: by targeting what the model says, we control what the model *is doing*

I wonder if theres a way to guess ahead of time whether or not a given activation is likely to be meaningfully semantically-interpretable.

Day: Fri, Apr 3 12026

Reward design and inference from language

Past work: have LLM give feedback in context on some trajectory, using LLM's classification (reward) modeling.

Eureka:

- Takes a natural language task description and environment source code and produces human-level reward model, can modify reward function in loop.
    - Isn't this kinda like what VLA models do? In the sense that VLA have a lot of intuitions of what humans think is good or bad, and they can be finetuned to directly access this reward modeling knowledge, to the point where if you can finetune it to a system, it can leverage its crazy priors to automatically generate actions given some natural language prompt.
        - Do VLAs use an internal reward function though? I'm not sure if its quite the same... perhaps in-layer reward functions...

Eureka:

- Vibe coded rewards is more fun (but gets less velocity?)
    - Eureka reward function iteration with human in the loop
        - Doesn't this kinda sound like disp?

Unsupervised Reinforcement Learning

- The *actual* solution!

Successor measures + Forward Backward representations

- Reminds me of autoencoders? All ML is compression?

A policy that predicts next state given past state and action

Language instruction -> Imagined trajectory (via video model)
"Zero-Shot language to behaviors without supervision"

Day: Wed, Apr 1 12026

Last class was midterm

RLHF, works well but requires extensive human data

- Why not RLHF but with RLAIF interspersed to make it more data efficient?

RLAIF works because discrimination and generation circuits in LLMs are distinct (simulating a good discriminator is different from simulating various kinds of simulation) RLAIF essentially connects them.

The future: fine-grainded preferences

Is anyone doing research on just taking like long LLM conversations, and at each conversation step, generate self-feedback, generate an alternative response and then simulate the user's response to classify.

Day: Wed, Mar 25 12026

Topic: Alignment Guarantees

So far: RLHF and pray

InstructGPT paper: SFT (to make initial behavior) -> RLHF (for finetuning behavior) -> Eval

How can we do something more principled and quantitative?

Verification is very difficult, lack of ground truth data.

- Satisficing? Mutual-modeling evolutionary theories of alignment?

Guarantee = Metric + Confidence + Assumptions

- Not a popular way of thinking (why?)

Metric = Return of policy under ground truth reward function
Confidence = Probabilistic bound on metric, e.g. 95% confidence bound on expected return
Assumptions =

Richard Sutton story - this is not perfect therefore it won't work (but its pragmatic???)

Guarantees spectrum

- Empirical, Probabilistic, Formal

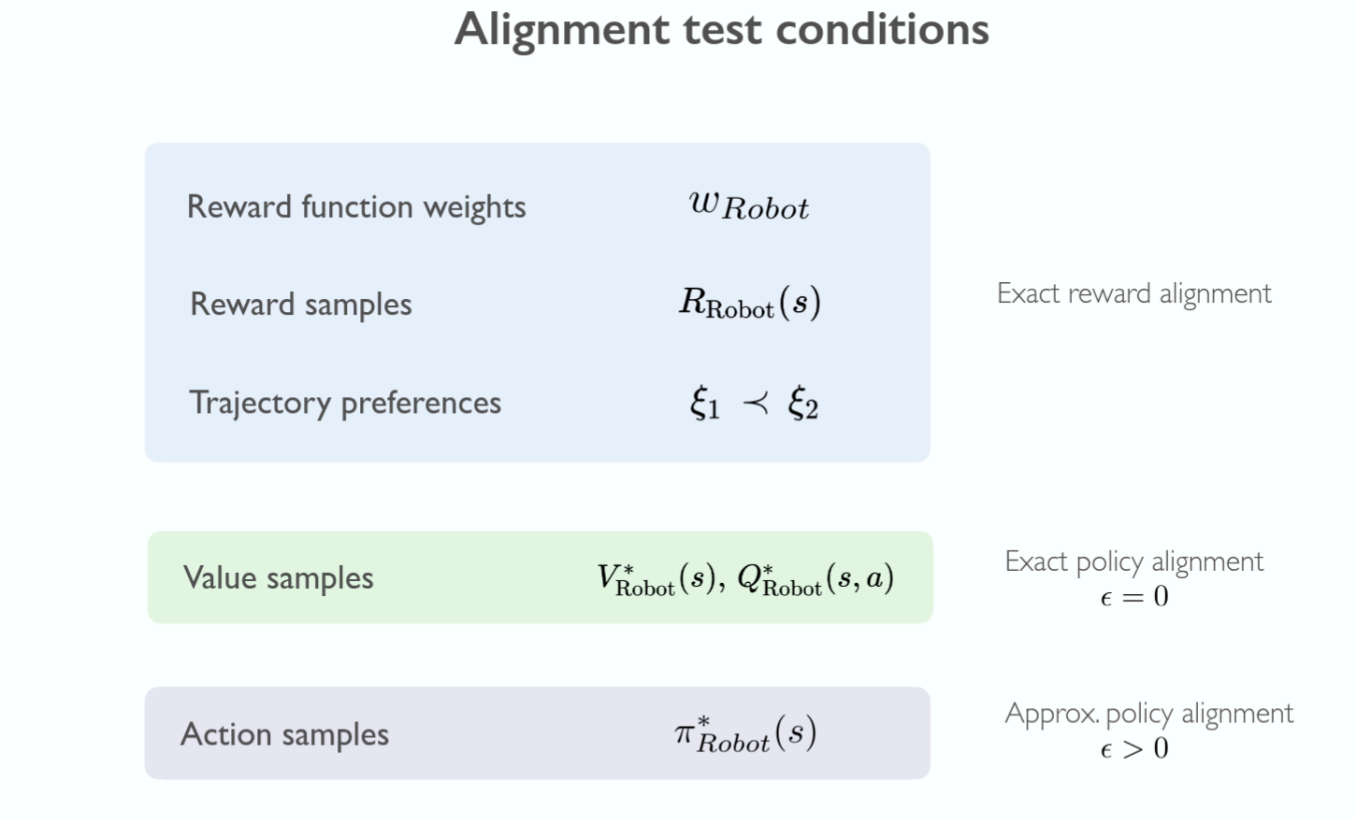
Goal: Strongest guarantees possible

"Can we design a small set of questions to ask an agent that verify alignment?"

- What if it lies?

Reward function halfspaces

- a "preference" between two different rollouts implies a partition in reward function space
- How?
- Can ask: is the reward function on the right side of the half-spaces in this reward function space?
  - Works if reward function is linear in features.
  - Can do this just via linear programming, figure the minimum set of half-space questions to fully define the full reward set (would this be questions with the number of features?)



Next steps:

- Make it work when features are unknown
- Make it work when preferences are noisy
    - Bradley-Terry model

Definition 1. *Given reward function R, policy π' is ϵ-**value aligned** in environment E if and only if*

$$V_R^*(s) - V_R^{\pi'}(s) \leq \epsilon, \forall s \in \mathcal{S}. \quad (1)$$

- Difference on state s for all states (S) value of policy difference with ideal is less than assumption bound epsilon.

Solution: Use MCMC instead of linear programming to make it work with ranked trajectory feature counts?

- Is there a fundamental relationship between MCMC and linear programming?

Paper: Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences

- https://arxiv.org/pdf/2002.09089

Day: Fri, Mar 13 12026

Direct Preference Optimization

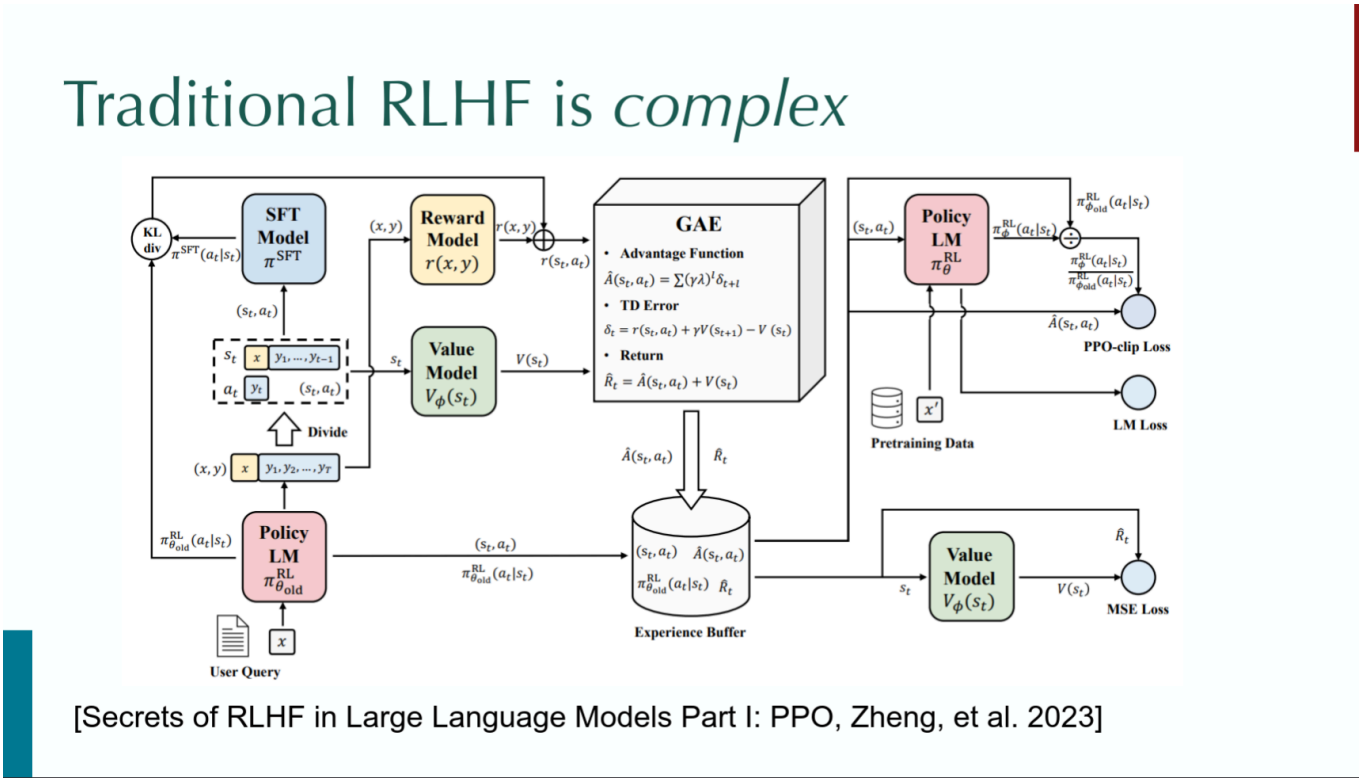
RLHF pipeline

- Unsupervised pre-training
- Supervisd fine-tuning on human demos
- Why RL at all? - want to go beyond

Why RL *after* SFT?

RLHF: KL regularization (want to keep KL to original model as close as possible)
Why KL regulatization?

- Old paper - information-theoretical -> loosing information from old distribution.



Solution: Direct Preference Optimization? (get rid of RL?)

$\pi_r^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

- π^* - ideal policy
- π_{ref} - reference finetuned model

It's a maxent policy.
Read more about it: https://cameronrwolfe.substack.com/p/direct-preference-optimization
Original paper: https://arxiv.org/pdf/2305.18290

Almost all models on OpnLLM leaderboard in 2024 used DPO.

- People are going back to not-direct methods

Part 2: Contrastive Preference Learning

Push up good policy, push down bad policy.

DPO is a special case of CPL - we learn a contextual bandit policy in the KL-constrained setting.

RLHF:

- Human feedback is difficult

Who's reward function?

- Does it matter in practice?

Long convo with claude about this: https://claude.ai/share/314655dd-186a-4ebd-b5f0-351678644acc

Day: Wed, Mar 11 12026

Today: RLHF: indirect methods (previous classes were cancelled, missed GAIL lecture)

Indirect methods: infer a reward function (vs direct methods = skip reward learning step)

T-REX paper - 2019, RLHF

GAIL - cannot significantly outperform the demonstrator

Standard assumption: expert is near-optimal

Ranked Trajectories $\tau_1, \tau_2 < \ldots < \tau_T$ -> T_REX -> reward function -> policy

Solution: Essentially just take a neural network that takes a trajectory and returns a score, where the cross-entropy loss penalizes mis-ordering of trajectories.

$$\hat{P}\left(J_{\theta(\tau_i)} < J_{\theta(\tau_j)}\right) = \frac{\exp \sum_{s \in \tau_j} r_{\hat{\theta}(s)}}{\exp \sum_{s \in \tau_i} r_{\hat{\theta}(s)} + \exp \sum_{s \in \tau_j} r_{\hat{\theta}(s)}}$$

Q: Why cross-entropy categorical learning vs ordinal regression? (Isn't ordinal regression kinda learning from messy feedback?)

Can do a lot of data aug from this. (subsampling, combining sampls) given a ranking, get more trajectories in the same rank.

T-REX compared to training from ground-truth

- It gets pretty close to ground truth policy performance (with explicit reward function) just by extrapolating reward function from demonstration.

D-REX: Auto-generated rankings

- Just add noise, more noise = worse ranking.

Bradley-Terry preference models

- Minimize regret

Reward Identifiability

Problems with preferences

- Everything
- https://openreview.net/forum?id=IL3sMH7cNc
- Preferences change over time, they aren't extrapolated for trajectory rankings
- Decoy preferences - can we empirically judge this?
  - Happens in sales, comparison between options
    - Does this still happen in binary choices?
- Circular preferences - How often does this happen?

Day: Fri, Feb 27 12026

Safe Imitation and Bayesian IRL

What does it mean for an agent to be 'safe'? (require minimal post-optimization)

- 'Formal safety' (never crash)
- Risk-sensitive safety (bounded value-at-risk)
- Robust safety (robust to out-of-distribution issues)

IRL review:

- Policy value under linear reward function (why do we assume linear reward function?)

Worst-case reasoning is the best we can do *if we don't know the ground-truth reward function* (but it should be over the data)

- 'Hoeffding method' (worst-case policy bounds relies on everything but the data itself)

MCMC

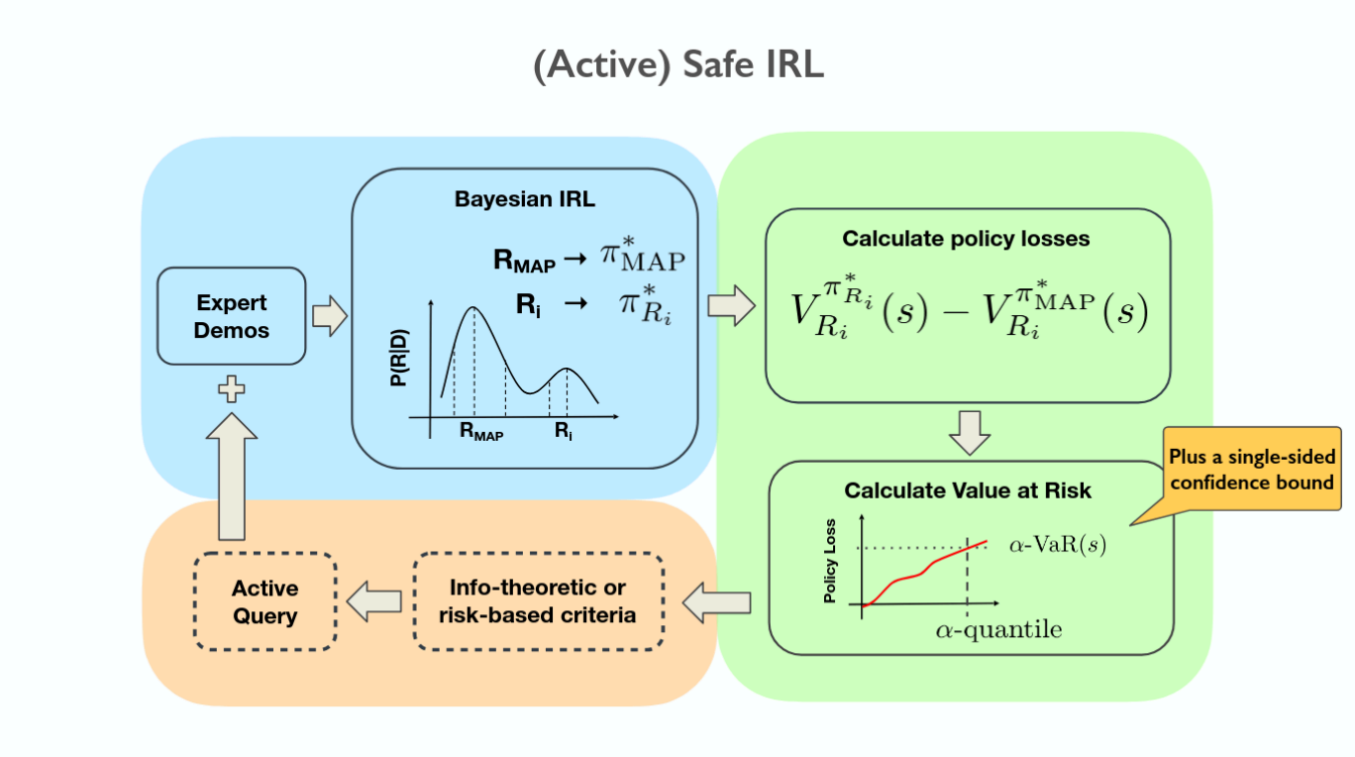
- Idea: rejection sampling is hard because you need a distribution that's always above your target distribution
- Solution: take a random walk around the distribution, use an *adaptive* Q!
  - Sample from a small point, fit a normal distribution, go to a new point based on sampled normal
- $P\left(x_{t+1} \mid x_1, \ldots, x_t\right) = P\left(x_{t+1} \mid x_t\right)$
- $P\left(x_{t+1}\right) = \sum_{x_t} P\left(x_{t+1} \mid x_t\right) P\left(x_t\right)$

Bayesian Inverse Reinforcement Learning (BIRL)

- Use MCMC to sample from posterior distribution $P\left(\mathrm{Reward} \mid \mathrm{Data}\right)$

Def: α-value at risk

- Given some investment strategy, you won't lose more than 500$ 95% of the time when using this investment strategy. (5% of time you lost more)
- How can we do this for imitation learning?
  - With probability $1-\delta$ no more than $\alpha\%$ of the outcomes will be worse than X
    - What is X exactly? A reward level? What actual outcomes does this correspond to?



- Compute optimal reward for MAP as well as n other samples R_i, calculate policies for all of them.
  - Calculate differences between policy losses (reward at state) R_i compared to ideal policy.

Active learning creates loop that reduces reward function uncertainty.

Okay this is incredibly cool.
The idea of distribution over reward functions allows you to give arbitrarily specific feedback to a robot / policy

- We can treat a review of past behavior, chopped up into good and bad behavior as positive and negative expert demonstration, which we can then use to adjust $P\left(\text{Reward} \middle| \text{Demos}\right)$ using our predictions $\left(\text{Demos} \mid \text{Reward}\right)$

Could we use this to improve RLHF? Specific RLHF? What about for reward maxmizing? LLM looks at reward trace, tries to identify where it went wrong, uses that as self-demonstration to

update the reward function, combine this with human reward function, unified rewards for goal optimization + human desires?

Wait this might actually be HUGE

Day: Wed, Feb 25 12026

Maximum Entropy IRL lecture.

IRL challenges

- RL in the inner loop
- Where do linear features come from?
- Infinite reward functions that explain behavior equally well, which one to choose? ("simplest"?)
- Suboptimal demonstrated behavior?
- Finite demonstrations, left out important conditions? (no demonstrations of crashes?)

Day: Fri, Feb 20 12026

Inverse reinforcement learning:

- Try to figure out what the reward function another agent is doing, so you can optimize for that learned reward function.

Naive Inverse Reinforcement Learning issues:

- Demonstrations are usually suboptimal, thus

Day: Wed, Feb 18 12026

Reward design is hard, demonstrations are hard

- What about feedback? (Human feedback? Self-feedback?)

How does clicker training work?

- You train an animal to do a behavior with training, how do deal with sudden stopping of reward causing behavior extinction?
    - Do reward occasionally and decreasing over time. Uncertainty of reward keeps behavior happening (because expectation is positive) but reward doesn't have to always occur.

RL agent attention should "cover" regions attended by human gaze

- Human gaze is a guide of where to look, but not end-all-be-all

- Scott's paper: The EMPATHIC Framework for Task Learning from Implicit Human Feedback
  - Coverage loss, guiding model on what to look at
    - %improvement up to 160%
    - T-REX vs T-REX + CGL (allows you to have the agent learn the ping-pong ball from following where the human is looking)

Day: Fri, Feb 13 12026

Today: Reward misspecification + hacking

What is reward misspecification?

- Where you have some idea of which worlds are better than others, and you try to write a function that can take any world and specify how good it is in such a way that creates an ordering, but your model of the world isn't detailed enough to create this function, and there are worlds that satisfy the function you wrote down, but not your intuition.
- Essentially the discrepancy between the true and the proxy reward function.
  - Solution: dynamic proxies?

Sparse vs Shaped/Dense

- Faster convergence -> trying to specify a value function at that point, value of each state

Q: Isn't the whole point of RL, to try and handle sparse rewards?

Potential-Based Reward Shaping

- Guaranteed not to effect policy ordering! (essentially providing priors)
- Equivalent to value function initialization: i assum this means the learned value functions do this automatically?

How bad is incomplete specification?

- Given a requirement of L attributes for true reward function, assuming even $L-1$ actually specified attributes, it can subtract an arbitrarily large amount from one of the unmentioned attributes, while gaining infinitesimally in one of the proxy attributes
  - Is it not possible to design reward functions where the model knows there might be other things to optimize for?
    - What if we just designed a reward function that rewarded models for trying to satisfy many other reward functions?

Reward hacking is sensitive to model size

- the more optimization power, the more severe the reward hacking

Reward function overfitting

- Designing reward functions by humans causes overfitting because we can't see all test cases
  - -> Just use sparse rewards?
- 'people are naturally drawn towards shaping' - i wonder if this is a psychological issue
- Hand-tuned reward functions are often customized to the optimization algorithm -> breaks ability to compare methods
- Invalidates ability of ablation studies

What to do?

- Don't use RL to learn your general algorithm from scratch, instead create a general algorithm, and specify it to your task (my hypothesis)

Day: Wed, Feb 11 12026

Today: Reinforcement Learning
Def: Learning through interaction with environment to maximize (discounted?) cumulative reward (given some reward function over states)

Goal: Learn a policy (learned mapping from states to actions)
Deterministic: $\pi : S \to A$, Stoachastic $\pi(a|S) = P(a|S)$ (what's the difference?)
Cumulative 'discounted' reward. (not optimizing directly for total reward?)

- Why discounted?
  - Infinite horizons are impossible to sum over, need a geometric series to get it to converge
  - Some training unstable without discounting (long horizons create large variance, especially if your reward function is not exact)

Goal: $J(\pi) = \mathbb{E}_{\pi, s_0 \sim d_0}[R_0]$ (R_0 = return starting at timestep 0)
'StrangeLoopSleuth: my temporal discount rate is vanishingly low' (how does this relate to cumulative discount factor?)

Key Challenges in RL:

- 'Credit Assignment Problem' (delayed effects, hidden mechanisms). did a bunch of actions, which actions were ideal? (humans suffer from this -> hendrick's cultural learning)
- Exploration vs Exploitation tradeoffs
- Non-stationarity & Distribution shift
  - Policy changes => state distribution changes

    - Note: neural networks chasing non-stationary distributions have trouble learning after a time ('they lose rank'? what does this mean?)
- Reward Design
    - Difficult to specifify (alignment problem!) -> reward hacking
        - Doing this manually means we have to work backward from conclusion
            - Ideally we should be able to do this automatically right? Directly from the really sparse total rewards? (chess?)
- Reproducibility & Sensistivity
    - Vulnerable to hyperparameter tuning and even hardware, comparing different methods is really difficult

Value functions (value of given state: $V(s)$, value of given action $Q(s, a)$)

Why don't we learn policies that try to optimize arbitrary reward functions? What does 'arbitrary reward functions' look like?

Optimal State Value: $V^*(s) = \max_\pi (V^\pi(s)) = \max_a Q^*(s, a)$
Optimal State-Action: $Q^*(s, a) = \max_\pi Q^\pi(s, a)$
'More easy to have an estimate of Q, can do policy-extraction just by picking best action'

- Can we not do the opposite? (why not?)
    - Need a world model to figure out next state from given action
        - Don't you kinda need this anyway? (i.e. doesn't Q need to have a model of the world?)

RL Algorithm Landscape

- Value function (critic-only) - deals poorly with partial observability, sample efficient
- Policy Search (actor-only) - high variance, low efficiency, robust to partial observability, bad features
- Actor-critic: best of both worlds

Specific Methods:

- Monte-carlo, estimate value by averaging episode returns
    - Every state-action pair visited, store in matrix, update expected reward
        - Given any state-action pair visited over all rollouts, get average
    - Unbiased, simple, high-variance, needs full episodes, huge state space, can't search space efficiently
- Bootstrapping - Q-Learning (off-policy)
    - 'off-policy' = can learn from any data at all, guaranteed to convert to optimal

    - $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a) \right]$
- Bootstrapping - SARSA (on-policy)
    - $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q\left(s', \pi\left(s'\right)\right) - Q(s,a) \right]$
- Sidenote: PPO (popular optimization strategy) *is on-policy* (which means its not particularly data-efficient)

Modern RL: Proximal Policy Optimization (PPO)

- Issue: Policy gradients can take too large of steps => unstable
- Idea: Constrain update size per update (via clipping rewards!)
- $L^{\text{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_{\hat{t}}, \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right) A_{\hat{t}}$
- Algo:
    - Collect trajectories using $\pi_{\theta_{\text{old}}}$
    - Compute advantages
    - Optimize clipped objective for K epochs
      Modern RL: Soft Actor-Critic (off-policy, sample efficiency)
- Key Idea: Maximize return & *Maximum entropy*, (you want the policy to perform as randomly as possible, while still doing well)
    - Idea: why entropy? maximimze exploration, be more robust, if one path is bad, you still are able to get to a reward signal
    - Objective: $J(\pi) = \sum_t \mathbb{E}\left[r\left(s_t, a_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid s_t\right)\right)\right]$
        - $\mathcal{H}(\pi) = -\mathbb{E}\left[\log \pi(a|s)\right]$
    - Pros: sample efficient, stable, strong on continuous control
    - Cons: more complex, continuous actions only, more hyperparameters
- Critic: Minimize soft bellman error.
- Actor:

Day: Fri, Feb 6 12026

Learn a policy π from expert π^* that we can query but who's cost (negative reward) function C we cannot directly observe.

Paper: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning
DAgger 'Dataset Aggregation'

- Run expert, collect trajectories, train policy on data (behavior cloning)
- Run learned policy, label states with what expert would have done at any given point, add to growing dataset
- Retrain on the dataset

Train policy on expert, test policy, learn away mistakes with expert guidance.

β-parameter shifting between copying the expert and being corrected by the expert.
Thoughts: Isn't this just what a baby does?

- Follow parents, slowly as policy comes online, shift to self-exploration + reinforcement with feedback.

Paper: Of Moments and Matching: A Game-Theoretic Framework for Closing the Imitation Gap
DAgger is an example of On-Q learning (you need an environment to learn in *and* an expert)
Alternative learning systems are:

- 'Reward moments' - do a full rollout, compare to chef trajectory across as many scoring functions as possible, if they are similar across many possible functions of the trajectories, they should in theory be similar in practice.
- 'Off-Q moments' - do a rollout, check chef's decision at every point, learn from action differences
- 'On-Q moments' (DAgger)

Day: Wed, Feb 4 12026

Required reading: Behavioral Cloning from Observation, 2018 UT Austin

Reinforcement Learning

- Reward over state space, figure out how to get the state with the maximum reward
    - Reward functions are usually sparse and don't have easy gradients to optimize over
        - Slow training, in practice intractable

Imitation Learning

- Existing observational data of agents doing things, not exact actions but some display of those actions
    - Need to learn to figure out what the actions were -> refine matching the display of the actions as best as possible.

Inverse Kinematics

- Recover reward function that was likely to create some demonstration when optimized

Why learn from demonstrations?

- 'The Perils of Trial-and-Error reward Design: Misdesign through Overfitting and Invalid Task Specifications'

How to provide demonstrations?

- 'Correspondence Problem' -> Isn't this just the aligment problem (the problem of defining a good reward function) but for robots?
- 'Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation'
  - Basically human controls a guide haptic thing that directly controls the robot
- Looking at video

Ideal robotics system idea (mine):

- Big worldmodel trained on video -> predict next autoencoded frame
- Physics-guided reward function (how physically likely is next frame?)
  - Harden the physics
  - This needs to be optimized
  - How necessary is this?
- Solid dataset across various robots
  - Can this be done with simulation?
    - Infer sensory stream
    - Infer what they see
    - Infer action
    - Train internal policy to match observation in simulated worldmodel

Sent to friend:

> thoughts on the idea that the solution to robots is just create really large world models (video models) and then extracting reward models and policy models from those giant world models? i.e. similar to extracting reasoning models from pre-trained language models?
>
> i.e. take a unified multimodel video-language transformer (predict next video/language token), if its been trained on videos of agents (read: humans), it should have learned an internal framework for figuring out what the most likely next action, or some notion of cause and effect (it should have learned an internal reward model for what makes a likely agent, as well as a policy model to predict what the agent is most likely to do). Lets say you then do a bunch of simulations of RL agents exploring a space or completing a game. You finetune the video transformer to take a rendering of the RL agents doing things and a trace of the RL agent's inputs and actions, it learns to predict inputs/outputs given a demonstration. You make sure to give it a short json description of the input and outputs of the agent
>
> Then you do normal 'generate me a video of a human doing something', 'ok, now take this picture of a robot and generate me a video of this robot doing the same thing', 'ok now

> generate me a trace of the robot's sensory inputs and motor outputs'
> (each of these would have to be RLHFd or something out of the model, and the last one would need the aformentioned simulation data)
>
> Boom, infinite dataset to do behavioral cloning from across a variety of robots, tasks, and environments. You could even then do RL on the world model itself and have it control a robot directly that you gave it a picture of, and map of its inputs and outputs, and a demonstration video of a task.

Okay

- Video-language-action model
- Demonstration video
  - Next-token prediction (done!)
    - Should learn model of underlying structure / environment (including agent!)
  - Prompt: what is agent seeing?
    - Examples third-person vs first-person (can be done physical environment, or virtual environment) -> generalized
  - Given demonstration video -> Predict full agent first person video (plausible)
  - Prompt: what is agent feeling / doing?
    - Theory: model should already have idea of what happens next, intuit ideas about reactions and all kinds of things. Model should *already have a partial behavioral reward function for the agent embedded inside it somewhere, it just needs to be elicited*
- Demonstration kinesthetic data
  - Recordings of human motion -> predict both 3rd person demonstration stream and motion sensors
  - Recordings of humans controlling robots -> Same
  - Recordings of robots exploring spaces -> Same, but also actions at any given point
- Reinforcement learning
  -

Friday class paper: 'A reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning'

Scaling up BC: pi_0.

- It can fold laundry!

Dynamics: $p\left(s_{t+1} \mid s_t, a_t\right)$

Day: Fri, Jan 30 12026

Prof: Scott Niekum

Prereqs:

- Supervise, Reinforcement learning
- NN/DL basics
- General ML concepts

Books: Sutton & Barto
Course structure: 50% lecture, 50% discussion

Alignment:

- Narrow view: specific classes of modern techniques (RLHF, mech interp, oversight) to make them in practice do what we want
- Broad view: any approach used by ML practitioners to express optimization objections

Building Blocks of alignment:
Data:

- Demonstration, feedback, multimodal signals (human approval signals)
  Algos:
- Behavior cloning
- Reward function design
- Inverse RL
  - RLHF

Why is alignment hard?

- **Reward Hacking**
  - (Measurement Problem!)

How do we give AI agents what individuals mean?

- Nuanced queues of facial expressions, universal signals of what is good/bad at a nuanced level?

Spurious Correlations

- Learns a proxy of success that almost, but not completely matches actual success
  - (Measurement Problem!)