DATA 37200: Learning, Decisions, and Limits
(Winter 2026)

Lectures 16-17: Modern Topics in Games and RL

Instructor: Frederic Koehler

Modern Topics in Games and RL

Part 1: Partial Information, CFR, Regret Matching

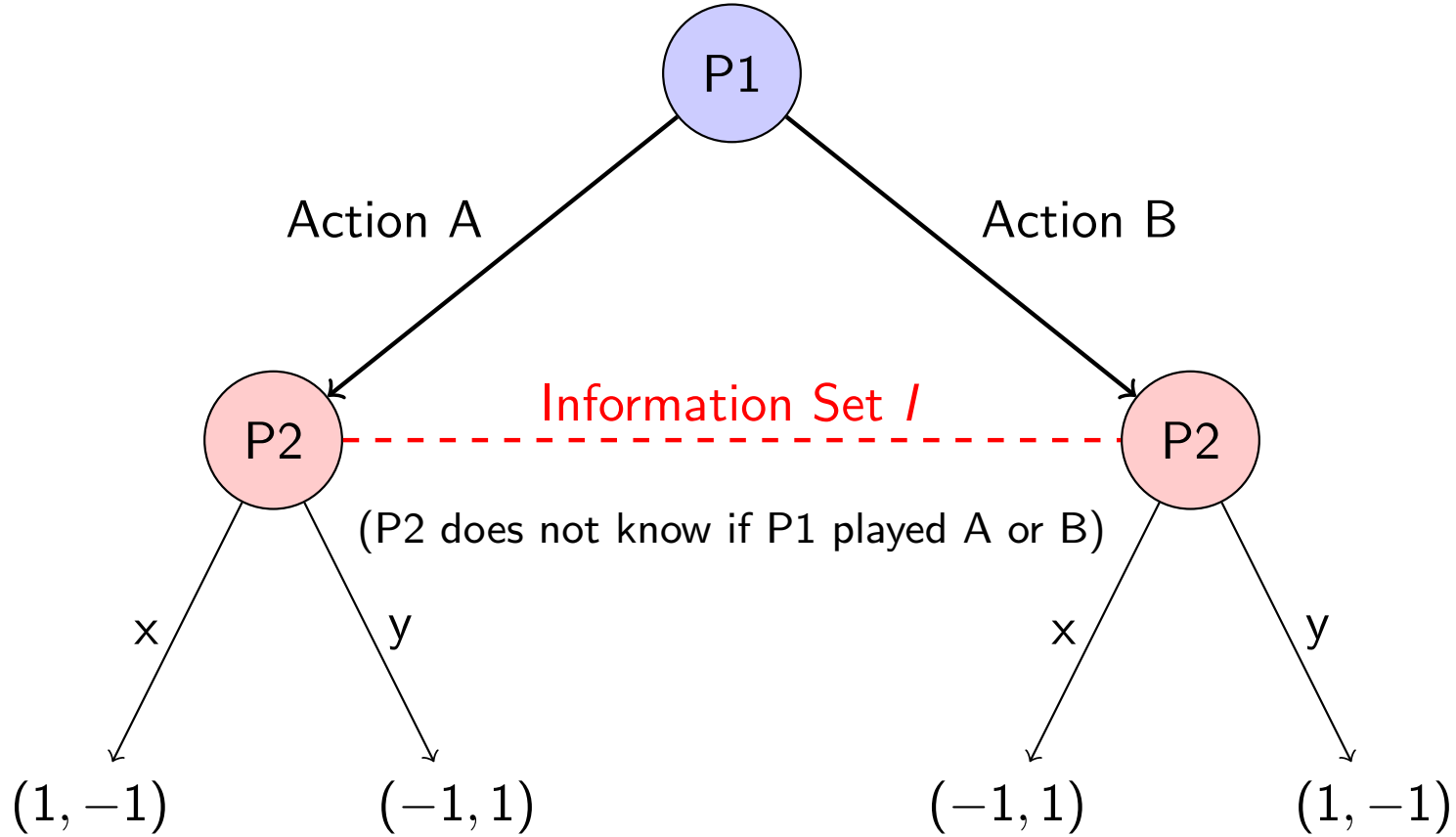
Part 2: Policy Gradients, PPO, and RLHF

Lecture 1: Games of Imperfect Information

Extensive-Form Games

- So far, we looked at normal-form (matrix) games. Many real games (Poker, Stratego) are sequential and have hidden information.
- **Game Tree:** Nodes are histories h, edges are actions a.
- **Information Sets (I):** A partition of nodes. If $h, h' \in I$, the player to act cannot distinguish between them (e.g., you know your cards, but not the opponent's).
- **Behavioral Strategy (σ):** A probability distribution over actions at each information set: $\sigma(a|I)$.

Visualizing Imperfect Information



- Player 2 observes the game state is in set I, but not the node. So P2 must choose the **same probability distribution** over actions $\{x, y\}$ at both nodes.
- **Strategy:** $\sigma_2(x|I)$ is a single value applied to the entire set.

Comparison to 2-player Markov game

We previously saw Markov games. How do extensive form games relate?

- Zero-sum: both Markov games and extensive form can be generalized beyond zero-sum. **But we will only consider zero-sum games today.**
- Rewards/payoffs: in MDP, rewards can be received at any time. In extensive form, payoffs are specified at the end of time.
- **Key difference:** information sets in extensive form game let us model partial information. (Analogous to partially observed MDP, i.e., POMDP.)

The Challenge of Extensive-Form Games

- A game tree can be exponentially large. In principle, can be converted to a huge matrix game, but this is impractical.
- **Goal:** Find a Nash Equilibrium in an efficient way.
- **Q:** Can we minimize regret locally at *each information set* instead of over the massive space of full strategies?

The Challenge of Extensive-Form Games

- A game tree can be exponentially large. In principle, can be converted to a huge matrix game, but this is impractical.
- **Goal:** Find a Nash Equilibrium in an efficient way.
- **Q:** Can we minimize regret locally at *each information set* instead of over the massive space of full strategies?
- **Remember:** In the full-information setting of zero-sum MDPs, we figured out how to solve this problem last class (VI reduces to solving local matrix games, matrix games solvable via EWA regret minimization).

The Challenge of Extensive-Form Games

- A game tree can be exponentially large. In principle, can be converted to a huge matrix game, but this is impractical.
- **Goal:** Find a Nash Equilibrium in an efficient way.
- **Q:** Can we minimize regret locally at *each information set* instead of over the massive space of full strategies?
- **Remember:** In the full-information setting of zero-sum MDPs, we figured out how to solve this problem last class (VI reduces to solving local matrix games, matrix games solvable via EWA regret minimization).
- **Counterfactual regret minimization:** A broadly similar strategy for partial information extensive form games. Very influential/successful in practice for poker etc.

Counterfactual Value

To define local regret, we need to define the value of an information set.

Reach Probabilities: Let $\pi^\sigma(h)$ be the probability of reaching history h under strategy profile σ.

- $\pi_i^\sigma(h)$: Probability player i's actions contributed to reaching h.
- $\pi_{-i}^\sigma(h)$: Probability the opponent's (and chance's) actions reached h.

Counterfactual Value $v_i(\sigma, I)$**:** The expected utility for player i given that information set I is reached, *assuming player i played to reach I*:

$$v_i(\sigma, I) = \sum_{h \in I} \pi_{-i}^\sigma(h) \sum_{z \in Z} \pi^\sigma(h \to z) u_i(z)$$

where z are terminal nodes and $u_i(z)$ is the utility.

Counterfactual Regret (CFR)

Counterfactual Regret of not taking action a at info set I at time t:

$$r_i^t(I, a) = v_i(\sigma^t_{I \to a}, I) - v_i(\sigma^t, I)$$

where $\sigma^t_{I \to a}$ is the strategy σ^t except at I, action a is played with probability 1.

Cumulative Counterfactual Regret:

$$R_{i,imm}^T(I, a) = \sum_{t=1}^{T} r_i^t(I, a)$$

The CFR Theorem (Theorem 3, Zinkevich et al., 2007): For every player i,

$$R_i(T) \leq \sum_{I_i} \max_a \max(0, R_{i,imm}^T(I, a)).$$

If a player minimizes cumulative counterfactual regret at *every* information set I, then their overall average regret in the full game is minimized.

The CFR Algorithm

Q: How do we minimize $R^T_{i,imm}(I, a)$ locally?

A: By choosing actions locally via a low-regret strategy such as exponential weights. Then we can guarantee

$$R_i(T) \leq \sum_{I_i} \max_a \max(0, R^T_{i,imm}(I, a)) = o(T).$$

In practice, rather than EWA, a "simpler" method called regret matching is used. We will discuss this next.

Convergence: Just like in normal-form games, if both players play CFR, the *average* profile $\bar{\sigma}^T$ converges to a Nash Equilibrium. This powered agents like **Cepheus** (Limit Hold'em) and **Libratus/Pluribus** (No-Limit Hold'em).

Regret Matching

Regret matching is a low-regret learning algorithm similar to EWA. It is not as appealing in terms of regret bound, but simple and *has no hyperparameters to optimize*.

Setup for Regret Matching

A variant of online forecasting:

- At each time step $t = 1, \ldots, T$, the environment assigns rewards $u_i(t) \in [0, 1]$ to each arm/expert $i \in \{1, \ldots, m\}$.

Setup for Regret Matching

A variant of online forecasting:

- At each time step $t = 1, \ldots, T$, the environment assigns rewards $u_i(t) \in [0, 1]$ to each arm/expert $i \in \{1, \ldots, m\}$.
- The learner selects an arm/expert $i(t)$ according to some distribution $p(t)$.

Setup for Regret Matching

A variant of online forecasting:

- At each time step $t = 1, \ldots, T$, the environment assigns rewards $u_i(t) \in [0, 1]$ to each arm/expert $i \in \{1, \ldots, m\}$.
- The learner selects an arm/expert $i(t)$ according to some distribution $p(t)$.
- The learner observes reward of all arms, and receives the reward $u_{i(t)}(t)$ for the selected arm $i(t)$.

Setup for Regret Matching

A variant of online forecasting:

- At each time step $t = 1, \ldots, T$, the environment assigns rewards $u_i(t) \in [0, 1]$ to each arm/expert $i \in \{1, \ldots, m\}$.
- The learner selects an arm/expert $i(t)$ according to some distribution $p(t)$.
- The learner observes reward of all arms, and receives the reward $u_{i(t)}(t)$ for the selected arm $i(t)$.

Remark: EWA in this setting is randomized: we sample expert i from the EWA distribution. We omit the details.

Setup for Regret Matching

A variant of online forecasting:

- At each time step $t = 1, \ldots, T$, the environment assigns rewards $u_i(t) \in [0, 1]$ to each arm/expert $i \in \{1, \ldots, m\}$.
- The learner selects an arm/expert $i(t)$ according to some distribution $p(t)$.
- The learner observes reward of all arms, and receives the reward $u_{i(t)}(t)$ for the selected arm $i(t)$.

Remark: EWA in this setting is randomized: we sample expert i from the EWA distribution. We omit the details.

Regret Definition: We define the cumulative regret against any fixed arm i as the difference between the rewards of i and our observed rewards:

$$R_i(T) = \sum_{t=1}^{T} \left(u_i(t) - u_{i(t)}(t) \right)$$

Our ultimate goal is to minimize the regret against the optimal arm in hindsight, i^*.

The Regret Matching Algorithm

Intuition: Track how much we "regret" not having played each action i in the past. If the regret is positive, play that action with probability proportional to the regret.

Algorithm (Hart and Mas-Colell, 2000):

1. Define the positive part of the regret:

$$R_i^+(T) = \max(0, R_i(T))$$

2. At time $t+1$, compute the total positive regret: $S(t) = \sum_{j=1}^{N} R_j^+(t)$.
3. If $S(t) > 0$, select arm $i(t+1)$ with probability:

$$p_i(t+1) = \frac{R_i^+(t)}{S(t)}$$

4. If $S(t) = 0$, pick an arm uniformly at random.

Theoretical Guarantee

Theorem (Regret Bound for Regret Matching)
If the learner uses the Regret Matching algorithm, then for any sequence of rewards $u_i(t) \in [0, 1]$, the expected regret against the optimal arm i^* satisfies:

$$\mathbb{E}[R_{i^*}(T)] \leq \sqrt{mT}$$

Proof Technique (Blackwell 1956): To prove this, we will track a potential function based on the sum of squared positive regrets:

$$\Phi_t = \sum_{i=1}^{N} (R_i^+(t))^2$$

We will show that the expected value of this potential grows at most linearly with time t.

Proof Part 1: Bounding the Step

Let the instantaneous regret for arm i at time t be:

$$r_i(t) = u_i(t) - u_{i(t)}(t)$$

Notice that $R_i(t) = R_i(t-1) + r_i(t)$.
We bound the squared positive regret:

$$\begin{aligned}(R_i^+(t))^2 &= \max(0, R_i(t-1) + r_i(t))^2 \\ &\leq (R_i^+(t-1) + r_i(t))^2 \\ &= (R_i^+(t-1))^2 + 2R_i^+(t-1)r_i(t) + (r_i(t))^2\end{aligned}$$

Summing over all arms i:

$$\Phi_t \leq \Phi_{t-1} + 2\sum_{i=1}^{m} R_i^+(t-1)r_i(t) + \sum_{i=1}^{m}(r_i(t))^2$$

Proof Part 2: Cancellation

Let us analyze the expected value of the cross term, conditional on the history up to time $t-1$. The selected arm $i(t) \sim p(t)$.

$$\mathbb{E}_{i(t)}\left[\sum_{i=1}^{m} R_i^+(t-1) r_i(t)\right] = \mathbb{E}_{i(t)}\left[\sum_{i=1}^{N} \left(S(t-1)p_i(t)\right) r_i(t)\right]$$

$$= S(t-1)\mathbb{E}_{i(t)}\left[\sum_{i=1}^{m} p_i(t)\big(u_i(t) - u_{i(t)}(t)\big)\right]$$

Now, look closely at the sum:

$$\sum_{i=1}^{m} p_i(t)u_i(t) - \sum_{i=1}^{N} p_i(t)u_{i(t)}(t)$$

The first term is exactly the expected reward $\mathbb{E}_{i(t)}[u_{i(t)}(t)]$. The second term pulls $u_{i(t)}(t)$ out of the sum over i, leaving $\sum p_i(t) = 1$. Thus, the inner sum is $\mathbb{E}_{i(t)}[u_{i(t)}(t)] - u_{i(t)}(t)$. Taking the expectation over $i(t)$ makes this identically 0!

Proof Part 3: Final Convergence

Because the cross term is zero in expectation, taking the full expectation yields:

$$\mathbb{E}[\Phi_t] \leq \mathbb{E}[\Phi_{t-1}] + \mathbb{E}\left[\sum_{i=1}^{m}(r_i(t))^2\right]$$

Since rewards are in $[0,1]$, the instantaneous regret $r_i(t) \in [-1,1]$, so $(r_i(t))^2 \leq 1$.

$$\mathbb{E}[\Phi_t] \leq \mathbb{E}[\Phi_{t-1}] + N$$

Telescoping this inequality from $t = 1$ to T (with $\Phi_0 = 0$):

$$\mathbb{E}[\Phi_T] \leq NT$$

Finally, we extract the regret for the optimal arm i^*:

$$(R_{i^*}^+(T))^2 \leq \sum_{i=1}^{m}(R_i^+(T))^2 = \Phi_T$$

Taking expectations and applying Jensen's Inequality:

$$(\mathbb{E}[R_{i^*}(T)])^2 \leq \mathbb{E}[(R_{i^*}^+(T))^2] \leq mT \implies \mathbb{E}[R_{i^*}(T)] \leq \sqrt{mT}$$

Conclusions for this section

- Since 90s, computers have beaten humans at chess using tree search based methods (DeepBlue).
- AlphaGo ('10s): beat humans at Go, a difficult fully-observed game. Used tree search guided by a convolutional network. Alpha-Go variant based on pure self-play.
- Later, AI has also beaten humans at challenging partial information games like poker and "diplomacy" (a classic strategy game, where players chat with each other about their strategy/to negotiate).
- Practical success of game theory and regret minimization paradigm.

Bonus content: proof of CFR theorem

Remark: a simpler result with a similar type of proof is the performance difference lemma for single player MDPs. (See Foster-Rakhlin notes.) A version of performance difference lemma shows up in the analysis below.

Technical note: for the following proof, assume that information sets do not cross different times. (This is because we are mostly interested in perfect recall games, where players remember what their previous moves were. This means they know how many times they moved.)

The Performance Difference Lemma (1)

Setup: Comparing Strategies over Time

We want to compare our current strategy σ against a reference strategy σ^* (e.g., the optimal strategy).

Let's define a sequence of **hybrid strategies** $\sigma^{(s)}$ for $s = 0, \ldots, D$ (where D is the max game depth):

- $\sigma^{(s)}$: Plays σ^* (reference) for steps $1, \ldots, s$.
- Then plays σ (current) for steps $s+1, \ldots, D$.

Observations:

- $\sigma^{(0)} = \sigma$ (Plays σ from the start).
- $\sigma^{(D)} = \sigma^*$ (Plays σ^* everywhere.)

Telescoping Sum:

$$u_i(\sigma^*) - u_i(\sigma) = \sum_{s=1}^{D} \left[u_i(\sigma^{(s)}) - u_i(\sigma^{(s-1)}) \right] = \sum_{s=1}^{D} \Delta_s$$

The Performance Difference Lemma (2)

Analyzing the Step Difference

Consider the difference $\Delta_s = u_i(\sigma^{(s)}) - u_i(\sigma^{(s-1)})$.

- These strategies are identical **before** time s (both play σ^*).
- They are identical **after** time s (both play σ).
- They differ **only at time** s:
  - $\sigma^{(s-1)}$ plays σ at time s.
  - $\sigma^{(s)}$ plays σ^* at time s.

The utility difference is the sum over all information sets I at depth s:

$$\Delta_s = \sum_{I \in \text{Depth}_s} \mathbb{P}(\text{reach } I | \sigma^{(s)}) \\ [\text{Value}(\text{play } \sigma^* \text{ then } \sigma, I) - \text{Value}(\text{play } \sigma \text{ then } \sigma, I)]$$

The Performance Difference Lemma (3)

Connecting to Counterfactuals

Let's break down the probability term:

$$\mathbb{P}(\text{reach } I|\sigma^{(s)}) = \sum_{h \in I} \pi_i^{\sigma^*}(h) \cdot \pi_{-i}(h)$$

(Because up to depth s, player i followed σ^*).
Now look at the value difference at h and let $I = I(h)$ be its information set. Since future play is fixed to σ:

- Value of playing σ at h: $v^{\text{cond}}(\sigma, h)$
- Value of playing σ^* at h: $\sum_a \sigma^*(a|I) v^{\text{cond}}(\sigma|_{I \to a}, h)$

Substituting this back (and absorbing $\pi_{-i}(h)$ into the value to get counterfactual value v_i):

$$\Delta_s = \sum_{h \in \text{Depth}_s} \pi_i^{\sigma^*}(h) \left(\sum_a \sigma^*(a|I) v_i(\sigma|_{I \to a}, h) - v_i(\sigma, h) \right)$$

The CFR Theorem Proof (Conclusion)

Summing it all up

Summing the differences over all time steps $s = 1 \dots D$:

$$u_i(\sigma^*) - u_i(\sigma) = \sum_{s=1}^{D} \Delta_s = \sum_h \pi_i^{\sigma^*}(h) \underbrace{\left(\sum_a \sigma^*(a|I) v_i(\sigma|_{I \to a}, h) - v_i(\sigma, h) \right)}_{\text{Instantaneous Regret against } \sigma^*}$$

Let $r_i(h, a) = v_i(\sigma|_{I \to a}, h) - v_i(\sigma, h)$.

$$u_i(\sigma^*) - u_i(\sigma) = \sum_h \pi_i^{\sigma^*}(h) \sum_a \sigma^*(a|I) r_i(h, a)$$

If σ^* is a best response (pure strategy), it selects one action a^* at each I, so using that $\pi_i^{\sigma^*}(h) \leq 1$:

$$R_i(T) \leq \sum_I \max_a \max \left(0, \sum_{t=1}^{T} r_i^t(I, a) \right) = \sum_I \max_a \max(0, R_{i,imm}^T(I))$$

■

A new topic: policy optimization and RLHF

- "Deep RL": combine reinforcement learning with modern ML methods to deal with very large state and action spaces.
- At very high-level, two general approaches: value iteration (e.g. Deep Q-Networks) or policy gradient (e.g. PPO).
- We will now introduce the latter approach and briefly explain its role in modern language models.
- As we will see, practical versions of policy gradient usually need to estimate value function, so arguably "hybrid" of both ideas.

Lecture 2: Policy Gradients, RLHF, etc.

Transition to Deep RL

- In massive environments (e.g., generating text, controlling robots), tabular representations fail. We use a parameterized policy $\pi_\theta(a|s)$ (e.g., a Neural Network).
- **Objective:** Maximize expected cumulative reward $J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$.

Policy Gradient Idea:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right]$$

where $\hat{A}_t$ is the Advantage function (total expected reward under current policy). Estimate via sampling trajectories (i.e. monte carlo).

Warning: need variance reduction!

Choice of "advantage" is important for monte carlo.

- In vanilla PG (REINFORCE), we use the return G_t as the weight:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \cdot G_{i,t}$$

  where Denote observed total reward as

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

- **Stochasticity:** The return $G_{i,t}$ depends on *all* future actions and random transitions.
- **Credit Assignment (?):** If $G_t = 100$, is that because a_t was good, or because a_{t+10} was amazing? Raw returns don't distinguish.
- **Result:** High variance $\rightarrow$ noisy gradients $\rightarrow$ unstable training.

In practice: We need to reduce the variance of the gradient estimator without introducing bias (or with controlled bias).

Lecture 2: Reducing Variance with Advantage

The Baseline Trick

- We can subtract a baseline $b(s_t)$ from the return without changing the expected gradient:

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (Q(s_t, a_t) - b(s_t))\right]$$

- **Why?** The baseline $b(s_t)$ is independent of the action a_t.

Defining Advantage $\hat{A}_t$ (idealized)

- The optimal baseline is the state value $V^\pi(s_t)$ (expected return from s_t).
- This yields the ideal Advantage Function:

$$\hat{A}(s_t, a_t) = \underbrace{Q^\pi(s_t, a_t)}_{\text{Result of taking } a_t} - \underbrace{V^\pi(s_t)}_{\text{Average result at } s_t}$$

- If $\hat{A}_t > 0$, action a_t was better than expected $\rightarrow$ reinforce it.
- If $\hat{A}_t < 0$, action a_t was worse than expected $\rightarrow$ discourage it.

Typically, train a second model to estimate value function.

Summary of this key part of the lecture

- Just naively using the returns and doing PG is considered a bad idea in most cases.
- Returns have too much variance/poor credit assignment.
- Real life: usually need a second model to estimate the value of state and fix this problem!
- These two models are often called "actor" and "critic".
- (Training critic is usually not complicated, we can just train it by having the actor play and the critic does regression to predict the reward that the actor gets playing from a state.)

Proximal Policy Optimization (PPO)

Another concern with Vanilla PG: Taking too large a step in θ space can collapse the policy. We want a "Trust Region", e.g. constrains policy to be close in KL. This idea led to TRPO and later PPO.

Proximal Policy Optimization (PPO)

Another concern with Vanilla PG: Taking too large a step in θ space can collapse the policy. We want a "Trust Region", e.g. constrains policy to be close in KL. This idea led to TRPO and later PPO.

PPO (Schulman et al., 2017) approximates TRPO by clipping the objective. Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ be the probability ratio.

At each step, update policy by maximizing the clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

Proximal Policy Optimization (PPO)

Another concern with Vanilla PG: Taking too large a step in θ space can collapse the policy. We want a "Trust Region", e.g. constrains policy to be close in KL. This idea led to TRPO and later PPO.

PPO (Schulman et al., 2017) approximates TRPO by clipping the objective. Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ be the probability ratio.

At each step, update policy by maximizing the clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

- Objective is pessimistic about excessively large changes to the policy. Note: objective lower bounds CPI objective $r_t(\theta)\hat{A}_t$.
- If $A_t > 0$ (good action), we increase $r_t(\theta)$ up to $1+\epsilon$.
- If $A_t < 0$ (bad action), we decrease $r_t(\theta)$ down to $1-\epsilon$.

This prevents destructively large policy updates and is standard in modern RL.

A crash course on LLMs

Since the age of GPT2, currently popular LLMs (Large Language Models) are autoregressive, so they try to generate language one token at a time (token: think something between a character and a word).

Main LLM training: train model to predict next word via cross-entropy/ multiclass logistic loss. Goal is minimizing perplexity, equivalently minimizing KL divergence between model and data. Makes a raw "smart" model but not user-friendly.

Reinforcement Learning from Human Feedback (RLHF)

To make the model more useful, the "base model" is further optimized using human preference data. (Change the objective from just predicting next word to answering questions etc.)

Problem: RL usually uses numerical rewards, but unclear how to elicit these from humans.

Reinforcement Learning from Human Feedback (RLHF)

To make the model more useful, the "base model" is further optimized using human preference data. (Change the objective from just predicting next word to answering questions etc.)
Problem: RL usually uses numerical rewards, but unclear how to elicit these from humans.

The RLHF Pipeline:

1. **Supervised Fine-Tuning (SFT):** Train π^{SFT} on high-quality human demonstrations.
2. **Reward Modeling (RM):** Collect human preference data (Prompt x, response y_1 is better than y_2). Train a reward model $r_\phi(x, y)$ using a Bradley-Terry loss (here $\sigma(z) = e^{-z}/(1 + e^{-z})$):

$$L(\phi) = -\log\left(\sigma(r_\phi(x, y_{win}) - r_\phi(x, y_{lose}))\right)$$

3. **RL Optimization:** Optimize the policy against r_ϕ (using PPO), while including an additional KL penalty to prevent drifting too far from π^{SFT}.

Direct Preference Optimization (DPO)

Concern with RLHF: Training a separate reward model and running PPO is unstable and computationally expensive.

DPO (Rafailov et al., 2023) bypasses the reward model. By analyzing the optimal policy equation for the KL-constrained RL problem, we can express the implicit reward *in terms of the policy itself*:

$$r(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}$$

Substitute this into the Bradley-Terry preference loss, and we can directly optimize the policy π_θ on human preference data using standard cross-entropy-style gradient descent—no PPO needed.

Derivation 1: The RLHF Objective

Step 1: The Objective

- In RLHF, we want to maximize rewards while staying close to the original model (π_{ref}) to prevent "mode collapse."
- Objective (KL-regularized reward):

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi}\left[r(x,y) - \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}\right]$$

Step 2: The Optimal Solution

- By a classic result (c.f. HW1 Problem 2) the optimizer π^* for this objective has a closed-form solution as a Gibbs measure:

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

- Where $Z(x) = \sum_y \pi_{ref}(y|x)\exp(\frac{1}{\beta}r(x,y))$ is a normalizing constant ($\log Z$ is a CGF).

Derivation 2: The "Implicit Reward"

Step 3: Solve for $r(x, y)$

- Take the log of both sides of the optimal policy equation:

$$\log \pi^*(y|x) = \log \pi_{ref}(y|x) + \frac{1}{\beta} r(x, y) - \log Z(x)$$

- Rearrange to isolate the reward function:

$$r(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{ref}(y|x)} + \beta \log Z(x)$$

Note:

- We have now expressed the reward strictly in terms of the optimal policy π^*, the reference π_{ref}, and a constant $Z(x)$.
- We treat our neural network π_θ as the "optimal policy" we are solving for.

Derivation 3: The DPO Update

Step 4: Bradley-Terry Preference Model

- The probability a human prefers winning response y_w over losing response y_l depends on the reward difference:

$$P(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l))$$

Step 5: Substitution and Cancellation

- Substitute our derived $r(x, y)$ into the difference:

$$r(x, y_w) - r(x, y_l) = \left[\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} + \beta \log Z(x)\right] - \left[\beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} + \beta \log Z(x)\right]$$

Result: A loss function dependent *only* on the policy π_θ!

Modern LLMs

- Semi-recent development in LLM world: using RL methods to teach the model how to "reason".
- Reasoning: spend more time + tokens on a problem, hopefully resulting in a higher quality answer, without need for explicit prompting (c.f. GPT3).
- Recommended reading: *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*
- Teach reasoning to LLMs using vaguely PPO-style RL method. GRPO, Group Relative Policy Optimization, in DeepSeek-R1.

Thanks!

- Hopefully you now have a better understanding of bandits, RL, control, game playing, etc than before the class.
- Remember to show up to final exam on **Tuesday**. You are allowed a one page, two-sided cheat sheet.
- I have posted some review problems from notes for study.
- Regret matching you should understand. But full details of CFR are not needed for exam. PPO/DPO/RLHF is not on the final exam.