The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

March 4, 2026 CSC 59866

Decision Making Algorithms for AI Agents like AlphaGo (RL + MCTS), TRPO, PPO, DPO, GRPO, StrateGo, and Intelligent AI Delegation

State of the Class

Recall: We discussed how basic Policy Gradients (REINFORCE) suffer from high variance, and how Actor-Critic methods attempt to stabilize learning.

Today's Goal: We will trace the evolution of state-of-the-art decision-making algorithms, starting with physical game board domination (AlphaGo) all the way to modern LLM alignment (DPO/GRPO) and Human-AI teaming.

Today's Agenda

Mastering Perfect & Imperfect Information Games

- AlphaGo
- StrateGo

The Evolution of Policy Optimization

- TRPO
- PPO

Aligning Agents Without Reward Models

- DPO
- GRPO

Mastering Perfect & Imperfect Information Games

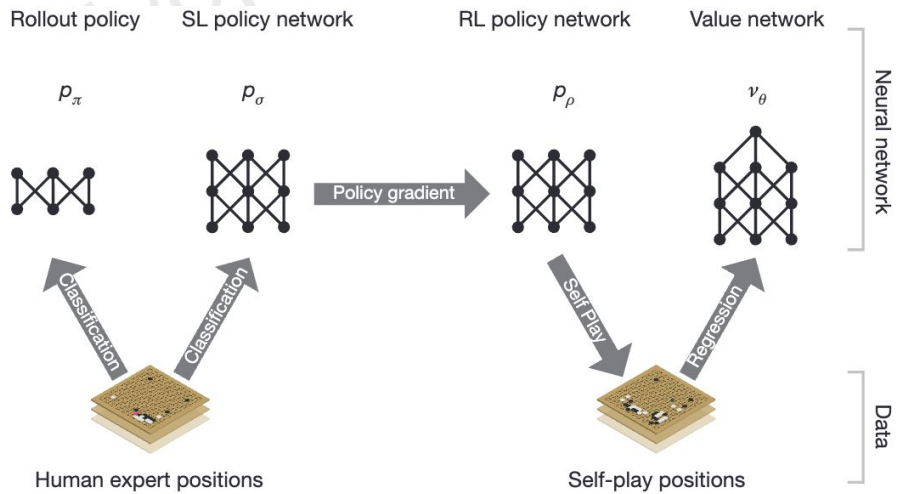
AlphaGo: RL + Monte Carlo Tree Search

Problem: Go has 10^{170} states; exhaustive search is impossible.

Policy Network: Narrows the search beam to high-probability moves.

Value Network: Truncates search depth by evaluating positions.

Rollouts: Fast simulations to estimate the "ground truth" value.



AlphaGo: Selection & MCTS Math

The agent selects actions by maximizing action value plus an exploration bonus:

$$a_t = \text{argmax}_a \left(Q(s,a) + u(s,a)\right)$$

Where the bonus $u(s,a)$ decays with visits to ensure convergence:

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

Q(s,a): expected action value
u(s,a): upper confidence bound exploration bonus
P(s,a): prior probability of that action (output by the policy network)
N(s,a): visit count for that node.

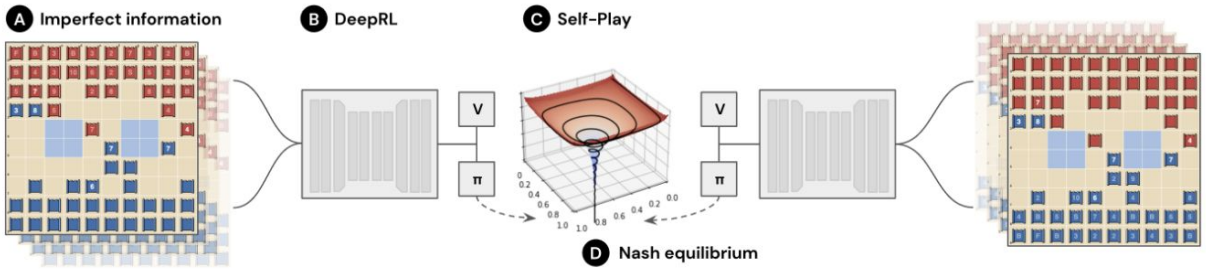
Stratego: Imperfect Information

Hidden Information: Unlike Go, piece types are private in Stratego.

Game Theory: Traditional search fails; requires finding a Nash Equilibrium.

Model-Free: Learns without building an explicit opponent belief model.

Tactics: Naturally learns bluffing and deception.



R-NaD: Replicator Dynamics

DeepNash uses Regularized Nash Dynamics to update policies without cycling:

$$\frac{d}{d\tau}\pi_\tau^i(a^i) = \pi_\tau^i(a^i)\left[Q_{\pi_\tau}^i(a^i) - \sum_{b^i}\pi_\tau^i(b^i)Q_{\pi_\tau}^i(b^i)\right]$$

Rewards are transformed to penalize moving away from the regularization policy:

$$r^i = r^i(a) - \eta\log\left(\frac{\pi^i(a^i)}{\pi_{\text{reg}}^i(a^i)}\right)$$

i represents the specific agent, tau represents the continuous time step or iteration, and a^i and b^i represent distinct actions. Eta is the learning rate and pi_reg is the regularization policy.

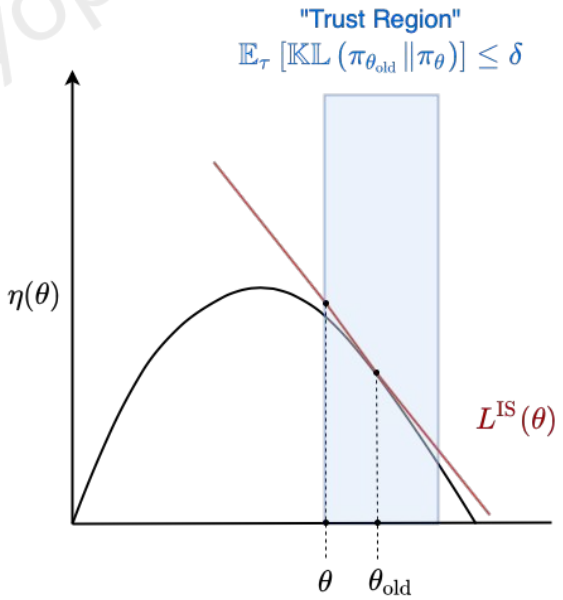
The Evolution of Policy Optimization

TRPO – Trust Region Policy Optimization

Problem: Vanilla policy gradients are unstable; bad updates destroy the policy.

Goal: Monotonic improvement at every single update step.

Trust Region: Only update parameters within a small "safe" neighborhood.



TRPO – Trust Region Policy Optimization

TRPO maximizes a surrogate objective L subject to a hard constraint on change:

$$\text{maximize}_{\theta} \quad L_{\theta_{\text{old}}}(\theta)$$

$$\text{subject to} \quad D_{KL}(\theta_{\text{old}}||\theta) \leq \delta$$

Delta is the maximum allowed threshold for the KL divergence.

Note: This requires computing the Hessian-vector product, which is mathematically robust but computationally expensive.

PPO – Proximal Policy Optimization

The "Trojan Horse" (Recall Lecture 7): This is the algorithm John Schulman used to turn NLP into RL!

Motivation: TRPO is complex to code and slow to run. PPO is a similar mechanism but less complex.

Solution: Use a first-order optimizer (like SGD) with a clipped ratio. Avoid walking off of “cliffs” with too large updates!

Impact: Better sample complexity and easier implementation.

Usage: The standard RL algorithm for OpenAI, DeepMind, and LLMs.

PPO – Proximal Policy Optimization

The objective is the minimum of unclipped and clipped ratios, creating a lower bound:

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

Where the probability ratio $r_t(\theta)$ is:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Aligning Agents Without Reward Models

RLHF - RL from Human Feedback

The Goal: Unsupervised LLMs just predict the next word. To make them act like helpful assistants (like ChatGPT), we must align them with human preferences.

Step 1: Supervised Fine-Tuning (SFT): First, train the base model on a high-quality dataset of instructions and ideal responses so it learns the basic format of a conversation.

Step 2: Reward Modeling (RM): Humans are given two model outputs for the same prompt and asked to choose the best one. We train a *second* neural network (the Reward Model) to predict these human preferences.

Step 3: RL Optimization: We use PPO to optimize the SFT model. The agent explores generating new text, the Reward Model scores it, and PPO updates the agent to maximize that score.

RLHF - RL from Human Feedback

To train the Reward Model (r_ϕ), we use the Bradley-Terry model. We want the reward for the winning response (y_w) to be higher than the losing response (y_l):

$$L_R = -\mathbb{E}_{(x,y_w,y_l)} \left[\log \sigma \left(r_\phi(x, y_w) - r_\phi(x, y_l)\right)\right]$$

Once the Reward Model is trained, we optimize the LLM policy (π_θ) to maximize the expected reward, minus a penalty:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta} \left[r_\phi(x, y) - \beta D_{KL}(\pi_\theta(y|x)||\pi_{\text{SFT}}(y|x))\right]$$

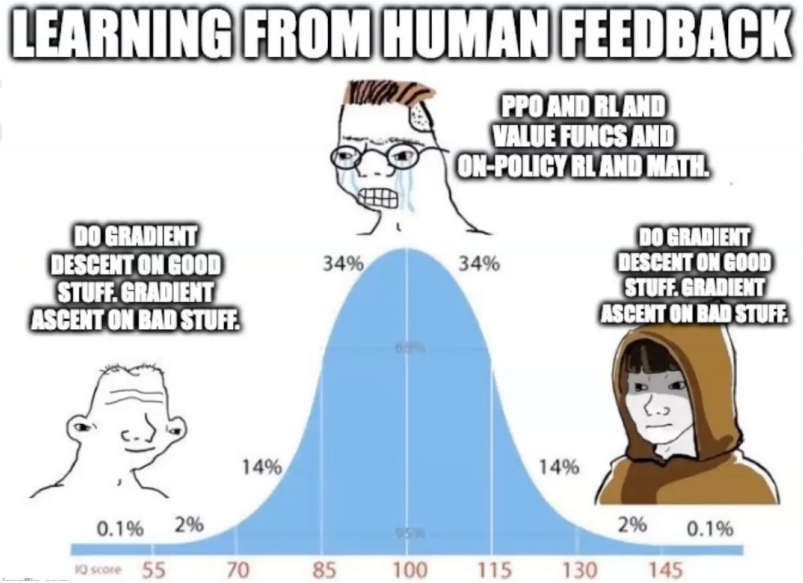
The KL Penalty (β): We force the new policy to stay close to the original π_{SFT} model. If we don't, the agent will find a way to "hack" the reward model by outputting high-scoring gibberish!

DPO – Direct Preference Optimization

Constraint: Standard RLHF requires training a 2nd massive "Reward Model".

Innovation: Your language model is *secretly* a reward model.

Benefit: No reinforcement learning loop is needed; just standard classification.



DPO – Direct Preference Optimization

DPO derives a simple cross-entropy loss from preference pairs (y_w, y_l) :

$$L_{DPO} = -\hat{\mathbb{E}}_{(x,y_w,y_l)} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right]$$

By increasing the probability of preferred response y_w relative to y_l, the model implicitly optimizes for human reward.

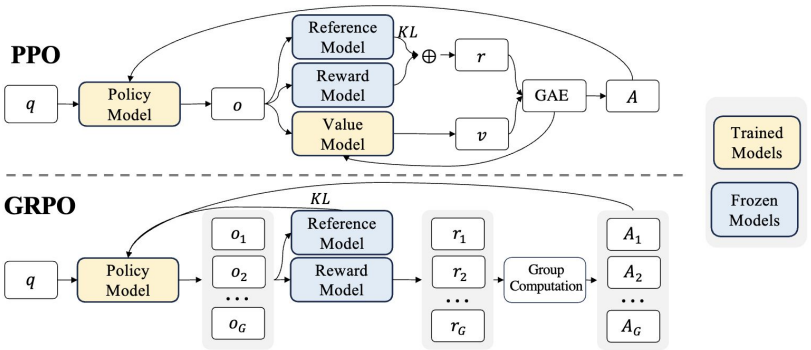
GRPO – Group Relative Policy Optimization

Memory Problem: PPO Actors and Critics take up massive VRAM.

Solution: Group Relative Policy Optimization.

Mechanism: Instead of a critic, use the average reward of a group of responses as a baseline.

Results: Powering DeepSeek's high mathematical reasoning scores.



GRPO – Group Relative Policy Optimization

GRPO samples G outputs and normalizes their rewards internally to calculate the advantage:

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

This eliminates the Value Model entirely, reducing training VRAM by roughly 50% for typical LLM training setups.

Human-Agent Interaction

Intelligent AI Delegation

The Context: As our agents (trained via PPO/DPO) become highly capable, we must decide *when* they should act independently and when they should ask humans for help.

The Core Problem: We must formally define the AI delegation problem to balance human control and AI autonomy.

The Trade-off: Intelligent delegation requires optimizing the strict trade-off between expected reward (task success) and human oversight (the cost of interrupting a human).

Connection to Senior Projects: If your project involves a user interface or an "AI Assistant," your methodology must account for when the agent is confident enough to execute an action versus when it flags the human for confirmation.

Intelligent AI Delegation

1. **Task Decomposition:** The "Delegator" (a human or a lead AI agent) breaks a complex, high-level objective into manageable, well-defined sub-tasks.
2. **Allocation & Capability Matching:** The system evaluates "Delegatees" (specialized AI agents or human experts) and assigns tasks based on the highest probability of safe success.
3. **Transfer of Authority:** The delegator establishes strict guardrails. It clearly defines the delegatee's operating boundaries, access rights, and accountability metrics.
4. **Execution & Trust Calibration:** As the delegatee executes the task, the delegator monitors its progress. The system dynamically updates "trust" in the delegatee based on results.
5. **Handoff or Intervention:** If the task succeeds, control and results are cleanly handed back. If the delegatee acts unpredictably or breaches a boundary, the delegator triggers a safe intervention protocol to take back control.

Questions?