Modeling Human Behavior Without Humans

Bringing Prospect Theory to Multi-Agent Reinforcement Learning

Sheyan Lalmohammed, Khush Gupta, Alok Shah

STAT 4830: Numerical Optimization for Data Science and ML

University of Pennsylvania

A Motivating Example

Let's play a game:

- I give you $50

A Motivating Example

Let's play a game:

- I give you $50
- We flip a fair coin. If heads, I give you $120; otherwise you get nothing.

A Motivating Example

Let's play a game:

- I give you \$50
- We flip a fair coin. If heads, I give you \$120; otherwise you get nothing.

Typical Reward-Maximizing Agent:

- Flips the coin — Expected value: \$60.

A Motivating Example

Let's play a game:

- I give you $50
- We flip a fair coin. If heads, I give you $120; otherwise you get nothing.

Typical Reward-Maximizing Agent:

- Flips the coin — Expected value: $60.

A Human:

- Likely prefers the sure $50.

A Motivating Example

Let's play a game:

- I give you $50
- We flip a fair coin. If heads, I give you $120; otherwise you get nothing.

Typical Reward-Maximizing Agent:

- Flips the coin — Expected value: $60.

A Human:

- Likely prefers the sure $50.

Agents Assume Rationality, Yet Humans Under Risk Aren't

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

Humans systematically deviate from rationality:

- **Risk Aversion**: Prefer sure gains over uncertain ones.

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

Humans systematically deviate from rationality:

- **Risk Aversion**: Prefer sure gains over uncertain ones.
- **Reference Dependence**: Judge outcomes relative to a baseline.

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

Humans systematically deviate from rationality:

- **Risk Aversion**: Prefer sure gains over uncertain ones.
- **Reference Dependence**: Judge outcomes relative to a baseline.
- **Probability Weighting**: Overweight small chances, underweight large ones.

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

Humans systematically deviate from rationality:

- **Risk Aversion**: Prefer sure gains over uncertain ones.
- **Reference Dependence**: Judge outcomes relative to a baseline.
- **Probability Weighting**: Overweight small chances, underweight large ones.
- All described by **Cumulative Prospect Theory (CPT)**.

The Problem

Traditional MARL agents:

- Optimize **expected cumulative rewards**.

Humans systematically deviate from rationality:

- **Risk Aversion**: Prefer sure gains over uncertain ones.
- **Reference Dependence**: Judge outcomes relative to a baseline.
- **Probability Weighting**: Overweight small chances, underweight large ones.
- All described by **Cumulative Prospect Theory (CPT)**.

Can we accomplish this without humans in the loop?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.
- Easier to predict, audit, and control agent decisions.

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.
- Easier to predict, audit, and control agent decisions.

Large Language Models:

- How do LLMs behave when interacting with other agents?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.
- Easier to predict, audit, and control agent decisions.

Large Language Models:

- How do LLMs behave when interacting with other agents?
- Can we give an LLM a Theory of Mind?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.
- Easier to predict, audit, and control agent decisions.

Large Language Models:

- How do LLMs behave when interacting with other agents?
- Can we give an LLM a Theory of Mind?
- Can CPT-style training unravel hidden dynamics in multi-agent LLM systems?

Why?

Simulating Human Behavior:

- Can agents learn another agent's utility function?
- Can they exploit human risk biases — e.g., offer deals that feel safe but are actually worse?
- Can we simulate negotiation, deception, and persuasion without real users?

Interpretability:

- Risk-sensitive behavior grounded in clear psychological models.
- Easier to predict, audit, and control agent decisions.

Large Language Models:

- How do LLMs behave when interacting with other agents?
- Can we give an LLM a Theory of Mind?
- Can CPT-style training unravel hidden dynamics in multi-agent LLM systems?
- *(Can it gaslight?)*

Our Definition of Success

1. Demonstrate that agents trained with CPT-adjusted rewards exhibit behaviors consistent with CPT principles.

Our Definition of Success

1. Demonstrate that agents trained with CPT-adjusted rewards exhibit behaviors consistent with CPT principles.
2. Show measurable differences in strategy optimization and reward dynamics between CPT and standard MARL agents.

Our Definition of Success

1. Demonstrate that agents trained with CPT-adjusted rewards exhibit behaviors consistent with CPT principles.
2. Show measurable differences in strategy optimization and reward dynamics between CPT and standard MARL agents.
3. Observe distinct risk-sensitive behaviors correlating with CPT parameterization.

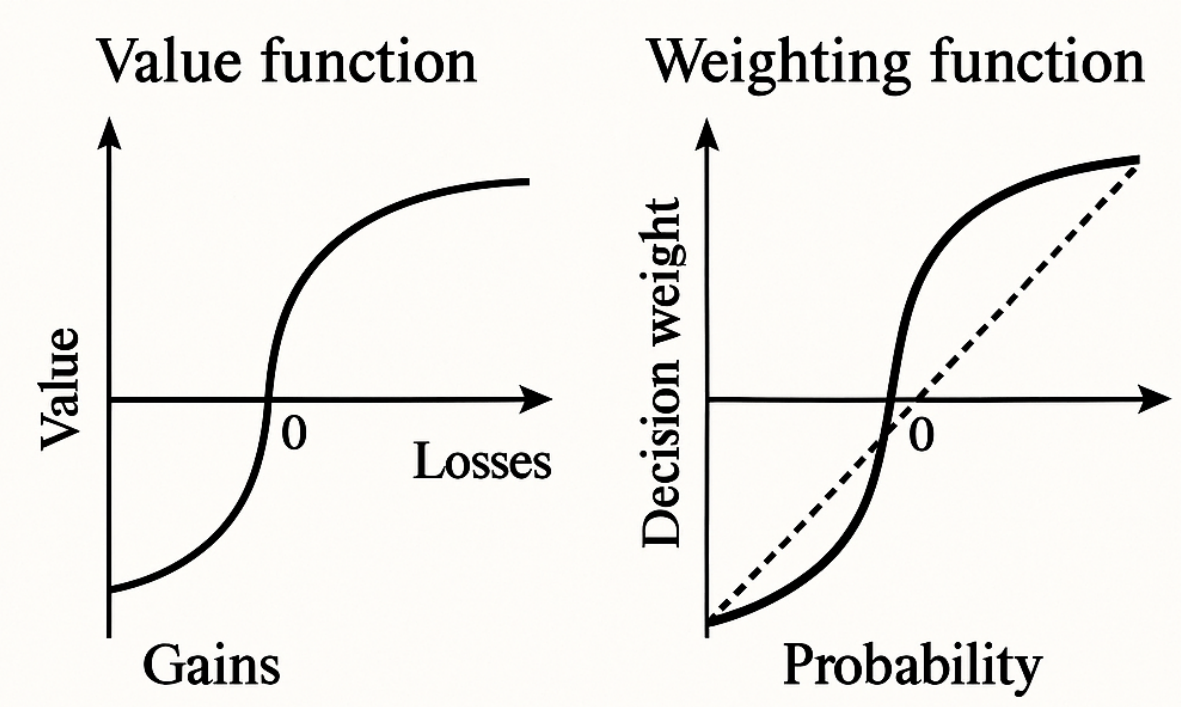
Our Definition of Success

1. Demonstrate that agents trained with CPT-adjusted rewards exhibit behaviors consistent with CPT principles.
2. Show measurable differences in strategy optimization and reward dynamics between CPT and standard MARL agents.
3. Observe distinct risk-sensitive behaviors correlating with CPT parameterization.
4. Successfully implement CPT within a MARL framework and achieve stable training.

Technical Approach

Literature Review - CPT Related

- **Prospect Theory (1979)**
  - Kahneman & Tversky introduce a *value function* (concave for gains, convex for losses) and an *inverse-S probability weighting* over outcomes.
- **Cumulative Prospect Theory (1992)**
  - Tversky & Kahneman extend PT to *rank-dependent* weighting of cumulative probabilities, handling multi-outcome gambles.



Literature Review - CPT MARL Related

- **Risk-sensitive RL Foundations**
  - Conditional Value at Risk (CVaR) optimization in MDPs (Bäuerle & Ott, 2014; Tamar et al., 2015)
  - Utility-based RL and exponential utility frameworks (García & Fernández, 2015)
- **Multi-Agent Risk-Aware RL**
  - CVaR-based objectives in cooperative MARL (Fan et al., 2019)
  - Entropy-regularized actor–critic as risk-sensitive MARL (Nachum et al., 2017)
  - *Risk-Sensitive Multi-Agent Reinforcement Learning in Network Aggregative Markov Games* (Chen et al., 2020)
- **CPT in Multi-Agent Settings**
  - *The Cumulative Prospect Theory and First Price Auctions* (Ewerhart & Leisen, 2010)
  - Integrating CPT utility into multi-agent actor–critic (recent CPT-MADDPG frameworks)

Modeling

Standard RL Objective:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Maximize expected (discounted) cumulative rewards.

Modeling

Standard RL Objective:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Maximize expected (discounted) cumulative rewards.

CPT Value Function:

$$C(X) = \int_{-\infty}^{0} w^-\big(P(u(X) > z)\big)\,dz - \int_{0}^{\infty} w^+\big(P(u(X) > z)\big)\,dz$$

Evaluate outcomes through subjective perception: nonlinear value $u(x)$ and distorted probabilities $w(p)$.

Modeling

Standard RL Objective:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

Maximize expected (discounted) cumulative rewards.

CPT Value Function:

$$C(X) = \int_{-\infty}^{0} w^-\big(P(u(X) > z)\big)\,dz - \int_{0}^{\infty} w^+\big(P(u(X) > z)\big)\,dz$$

Evaluate outcomes through subjective perception: nonlinear value $u(x)$ and distorted probabilities $w(p)$.

CPT-based RL Objective:

$$\max_\theta \mathbb{E}_{\pi_\theta}[C(R)]$$

Now, the agent optimizes for the *subjective* value of returns.

CPT as a "Wrapper" on Total Return

$$R = \sum_{t=0}^{T} r_t \quad \xrightarrow{\text{CPT steps}} \quad C(R)$$

- **Step 1: Subjective Value Transformation** ($u(x)$) Map outcomes through a nonlinear value function:
  - Losses hurt more than gains (loss aversion).
  - Sensitivity diminishes for extreme outcomes (diminishing sensitivity).

CPT as a "Wrapper" on Total Return

$$R = \sum_{t=0}^{T} r_t \quad \xrightarrow{\text{CPT steps}} \quad C(R)$$

- **Step 1: Subjective Value Transformation** ($u(x)$) Map outcomes through a nonlinear value function:
  - Losses hurt more than gains (loss aversion).
  - Sensitivity diminishes for extreme outcomes (diminishing sensitivity).

- **Step 2: Probability Distortion** ($w(p)$) Distort the likelihood of outcomes

CPT as a "Wrapper" on Total Return

$$R = \sum_{t=0}^{T} r_t \quad \xrightarrow{\text{CPT steps}} \quad C(R)$$

- **Step 1: Subjective Value Transformation** ($u(x)$) Map outcomes through a nonlinear value function:
  - Losses hurt more than gains (loss aversion).
  - Sensitivity diminishes for extreme outcomes (diminishing sensitivity).
- **Step 2: Probability Distortion** ($w(p)$) Distort the likelihood of outcomes
- **Step 3: Aggregation into** $C(R)$ Combine transformed values and distorted probabilities to compute the final subjective score.

Training Objectives: Critic and Actor

Critic (Q_ϕ): Evaluates actions.

- Predicts the expected future return (value) of actions.
- Goal: Minimize Bellman (Prediction) errors.

$$L(\phi) = \mathbb{E}_{\mathcal{D}}\left[(\text{predicted value} - \text{target value})^2\right]$$

Target value = immediate reward + discounted future value

Training Objectives: Critic and Actor

Critic (Q_ϕ): Evaluates actions.

- Predicts the expected future return (value) of actions.
- Goal: Minimize Bellman (Prediction) errors.

$$L(\phi) = \mathbb{E}_{\mathcal{D}}\left[(\text{predicted value} - \text{target value})^2\right]$$

Target value = immediate reward + discounted future value

Actor (μ_θ): Chooses actions.

- Picks actions that maximize CPT-adjusted returns.
- Goal: Maximize subjective outcomes based on critic's estimates.

$$L(\theta) = -\mathbb{E}_{\mathcal{D}}\left[\text{CPT-weighted } Q_\phi(\mathbf{o}, \mu_\theta(\mathbf{o}))\right]$$

CPT-weighting emphasizes more impactful outcomes (rare gains/losses).

How Do We Optimize Complex Objectives?

Many RL and CPT objectives involve difficult expectations:

$$\max_{\theta} \mathbb{E}[C(R)] \qquad \text{or} \qquad \nabla_{\theta} \mathbb{E}[C(R)]$$

How Do We Optimize Complex Objectives?

Many RL and CPT objectives involve difficult expectations:

$$\max_{\theta} \mathbb{E}[C(R)] \qquad \text{or} \qquad \nabla_\theta \mathbb{E}[C(R)]$$

Key Idea: Approximate Expectations by Sampling

- Collect trajectories $(\mathbf{o}, \mathbf{a}, r, \mathbf{o}')$ into a replay buffer $\mathcal{D}$.
- Sample mini-batches from $\mathcal{D}$ to estimate gradients:

$$\nabla_\theta J \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \, C(R^{(i)})$$

- Use these sampled gradients to update parameters with stochastic gradient descent (SGD or Adam).

How Do We Optimize Complex Objectives?

Many RL and CPT objectives involve difficult expectations:

$$\max_{\theta} \mathbb{E}[C(R)] \qquad \text{or} \qquad \nabla_\theta \mathbb{E}[C(R)]$$

Key Idea: Approximate Expectations by Sampling

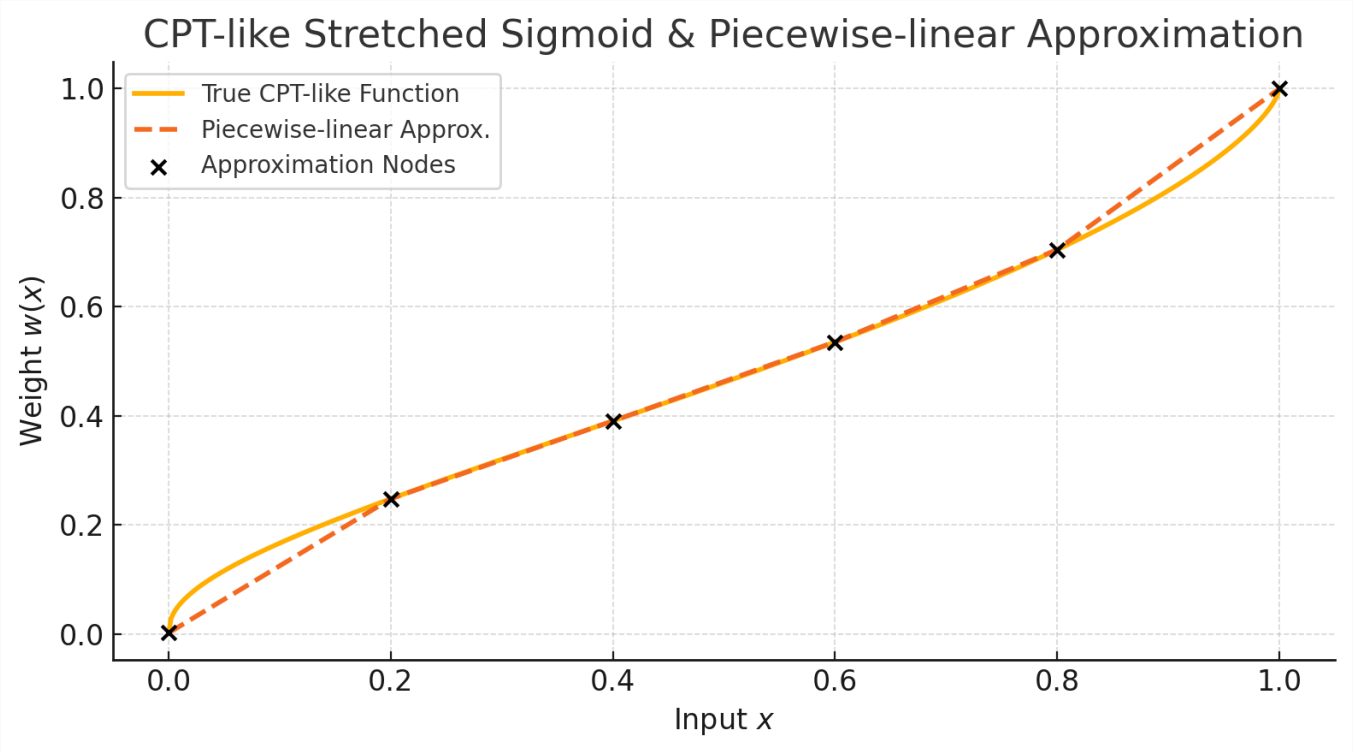
- Collect trajectories $(\mathbf{o}, \mathbf{a}, r, \mathbf{o}')$ into a replay buffer $\mathcal{D}$.
- Sample mini-batches from $\mathcal{D}$ to estimate gradients:

$$\nabla_\theta J \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \, C(R^{(i)})$$

- Use these sampled gradients to update parameters with stochastic gradient descent (SGD or Adam).

Intuition: Complicated RL integrals $\rightarrow$ Simple averages over finite batches of stored experience.

Better Approximation to CPT



True "stretched-sigmoid" CPT-like weighting (solid) vs. 6-segment piecewise-linear approximation (dashed), with nodes marked.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
    - **Actor Network:** Chooses actions from local observations.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).
- **Key Hyperparameters:**

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).
- **Key Hyperparameters:**
  - **Learning Rate ($\eta = 10^{-4}$):** Step-size for network updates.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).
- **Key Hyperparameters:**
  - **Learning Rate ($\eta = 10^{-4}$):** Step-size for network updates.
  - **Discount Factor ($\gamma = 0.99$):** Immediate vs. future rewards.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).
- **Key Hyperparameters:**
  - **Learning Rate ($\eta = 10^{-4}$):** Step-size for network updates.
  - **Discount Factor ($\gamma = 0.99$):** Immediate vs. future rewards.
  - **Target Update Rate ($\tau = 0.01$):** Stability of training.

Optimization Algorithm & Key Choices

- **Multi-Agent Deep Deterministic Policy Gradient** Each agent trains:
  - **Actor Network:** Chooses actions from local observations.
  - **Critic Network:** Evaluates actions using all agents' observations/actions.
- **Centralized Training, Decentralized Execution (CTDE)** During training, the critic uses global information for stability. At execution, actors use only local observations.
- **(CPT)-Adjusted Updates** Policy gradient updates emphasize actions leading to outcomes that are subjectively preferred (higher CPT value).
- **Key Hyperparameters:**
  - **Learning Rate ($\eta = 10^{-4}$):** Step-size for network updates.
  - **Discount Factor ($\gamma = 0.99$):** Immediate vs. future rewards.
  - **Target Update Rate ($\tau = 0.01$):** Stability of training.
  - **CPT Scaling ($\beta = 1.0$):** Emphasis on rare outcomes.

Implementation

Core Libraries: PyTorch, TorchRL, Vmas.

Key TorchRL Components:

- Data: TensorDict, ReplayBuffer, SyncDataCollector.
- Networks: TensorDictModule, MultiAgentMLP, ProbabilisticActor.
- Objectives: LossModule, TD0Estimator, SoftUpdate.

Custom CPT Code:

- CPTDDPGLoss(LossModule): Custom loss class.
- compute_cpt_integral(): Calculates $\Phi(\mathcal{R}_{batch})$.
- u_plus(), u_minus(), w_approx(): CPT component functions.

Issues

In order of most annoying (10+ hours debugging faulty code) to least annoying (needed to read more or learn more about a topic):

1. Reinforcement Learning
2. Hyperparameter tuning. Figuring out if we should be tweaking typical RL hyperparameters, CPT hyperparameters, or both. Whenever we would tweak some hyperparameters, we commonly kept overshooting the relevant values.
3. Getting a better approximation to our CPT objective function (We had to use a codebase which was completely disorganized and whose code only partially worked in our context).
4. CPT Agents rewards being too unstable. Massive Jumps - had to figure out how to smooth them.

Results

Original Analysis

We used three environments in our analysis:

1. **Competitive MPE** (Petting Zoo Simple Tag)
    - **Objective**: Predators work to "tag" or catch the prey, while the prey's goal is to evade capture.
    - **Rewards**: Rewards are structured so that predators gain rewards when they successfully tag the prey, and the prey receives a penalty when caught.
2. **Cooperative MPE** (Petting Zoo Simple Spread)
    - **Objective**: The agents work cooperatively to cover all the landmarks. Their goal is to position themselves so that each landmark is "covered" by at least one agent, maximizing overall performance.
    - **Rewards**: Rewards encourage efficient coverage of landmarks while also penalizing agents for collisions with one another, which promotes coordinated movement and spacing.

Extension

Competitive/Cooperative First Price Auction

- **Objective**: Each agent (bidder) submits a sealed bid; the highest bidder wins the item and pays their bid. In the competitive setting, each agent maximizes its own payoff; in the cooperative setting, agents coordinate their bids to maximize total group surplus.
- **Rewards**:
  - *Competitive mode*: The winning agent with valuation v and bid b receives reward

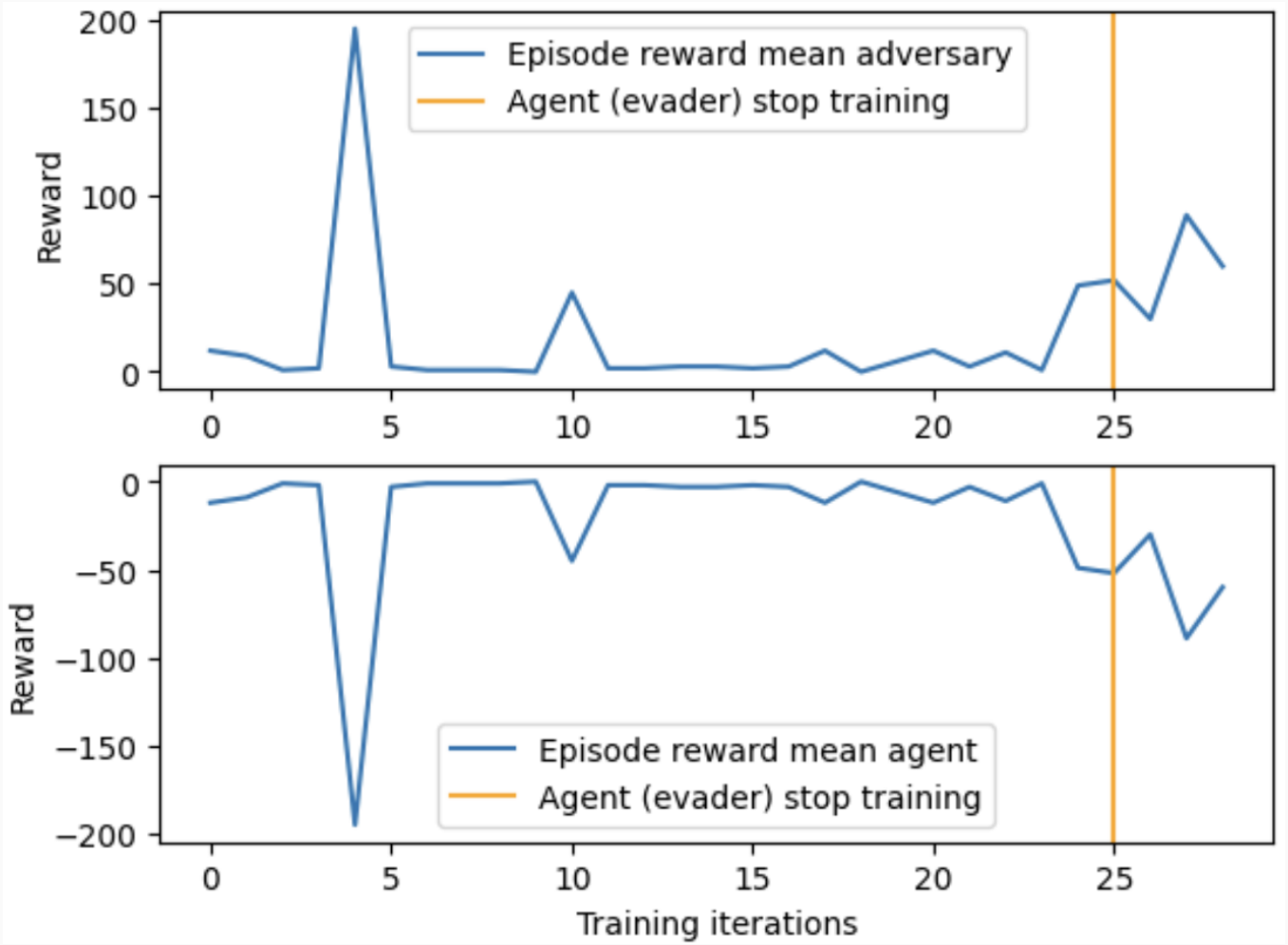
    $$v - b,$$

    and all losing agents receive reward 0.
  - *Cooperative mode*: The group's reward is the sum of all individual payoffs $\sum_i (v_i - b_i)$, which can then be shared equally or via a predefined team allocation.

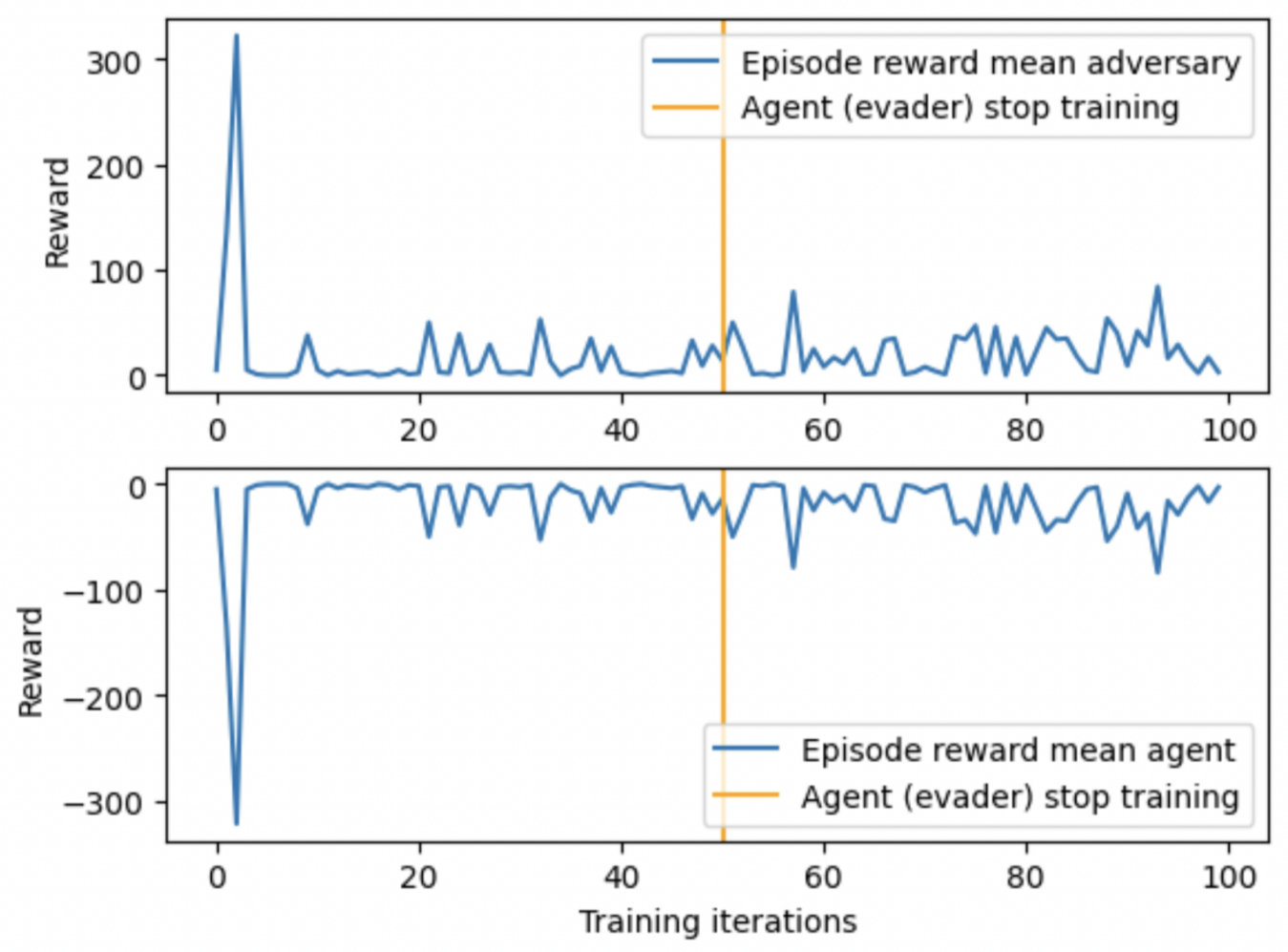
What happens when we put two CPT-trained agents against each other in Simple Tag?

- **Base Case:** Agents converge to a stable equilibrium reward (neither agent gets much gain)
- **Moderate Risk Seeking:** Agents still converge generally to a stable equilibrium with not much reward but there is variance once getting there.
- **Extreme Risk Aversion:** Agents no longer converge, the adversary guaranteed to catch the agent (the adversary doesn't take risks to increase gains, general fluctuations around some equilibrium)

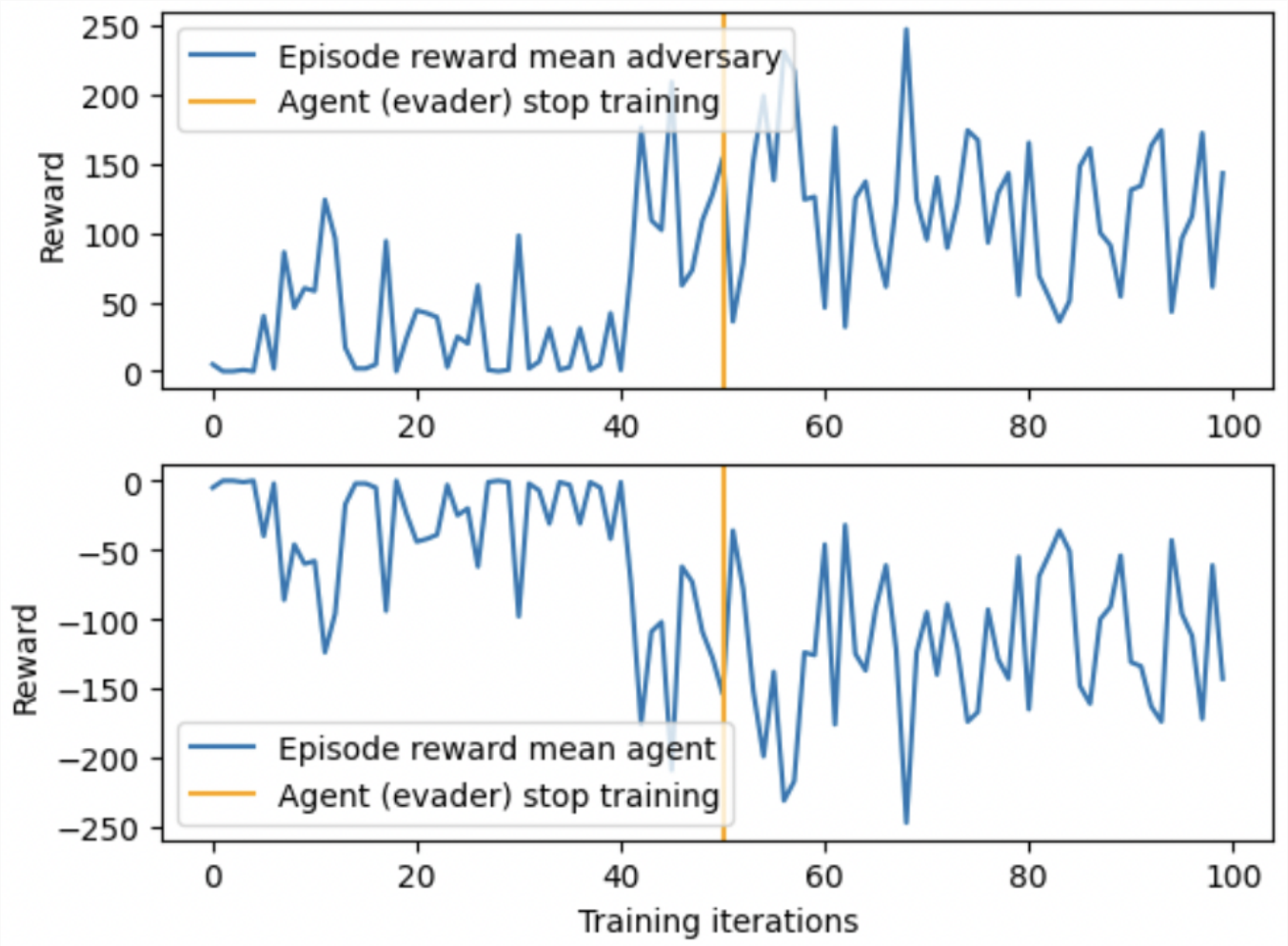
Competitive Environment – Baseline Reward



Competitive Environment – Moderate CPT (Risk Seeking)



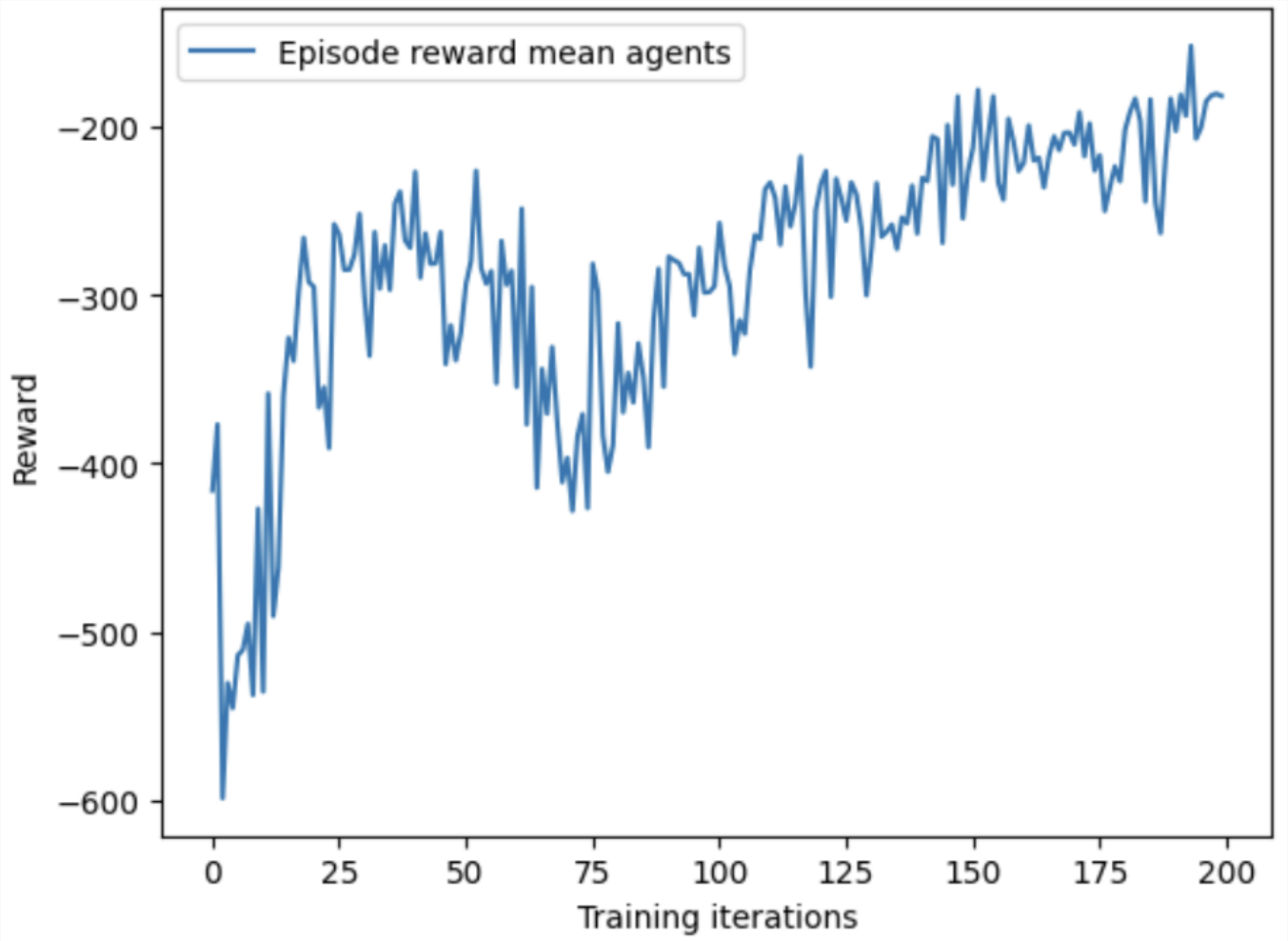
Competitive Environment – Extreme CPT (Risk Averse)



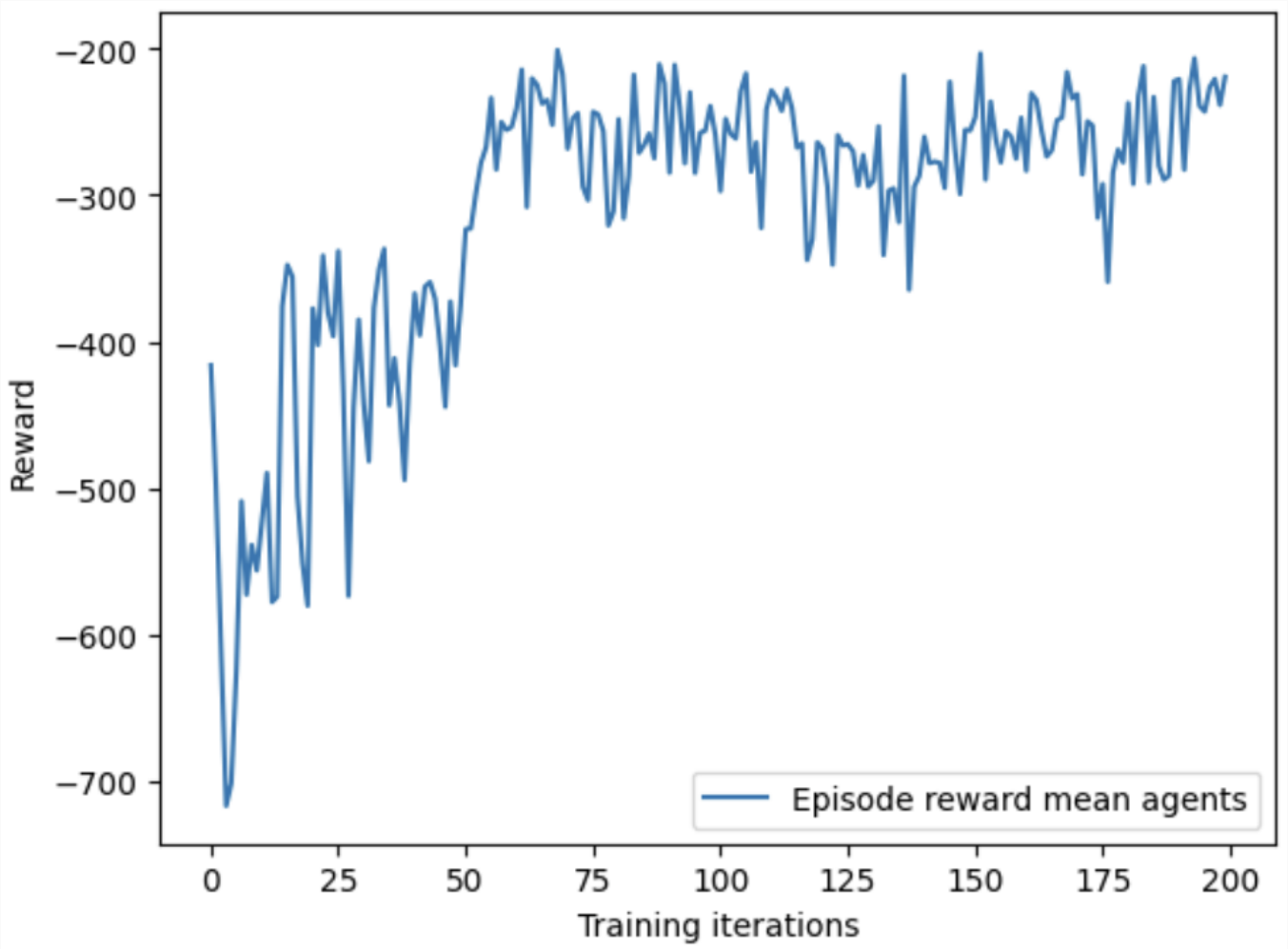
What happens when we put two CPT-trained agents to work with each other in Simple Spread?

- **Base Case:** Agents converge to a stable equilibrium reward
- **Moderate Risk Aversion:** Agents still converge (somewhat quicker) to a stable equilibrium but fluctuate actions heavily once reaching there.
- **Extreme Risk Aversion:** Agents get to original stable equilibrium reward value, but risk aversion to possibly hitting one another causes them to retract their position.

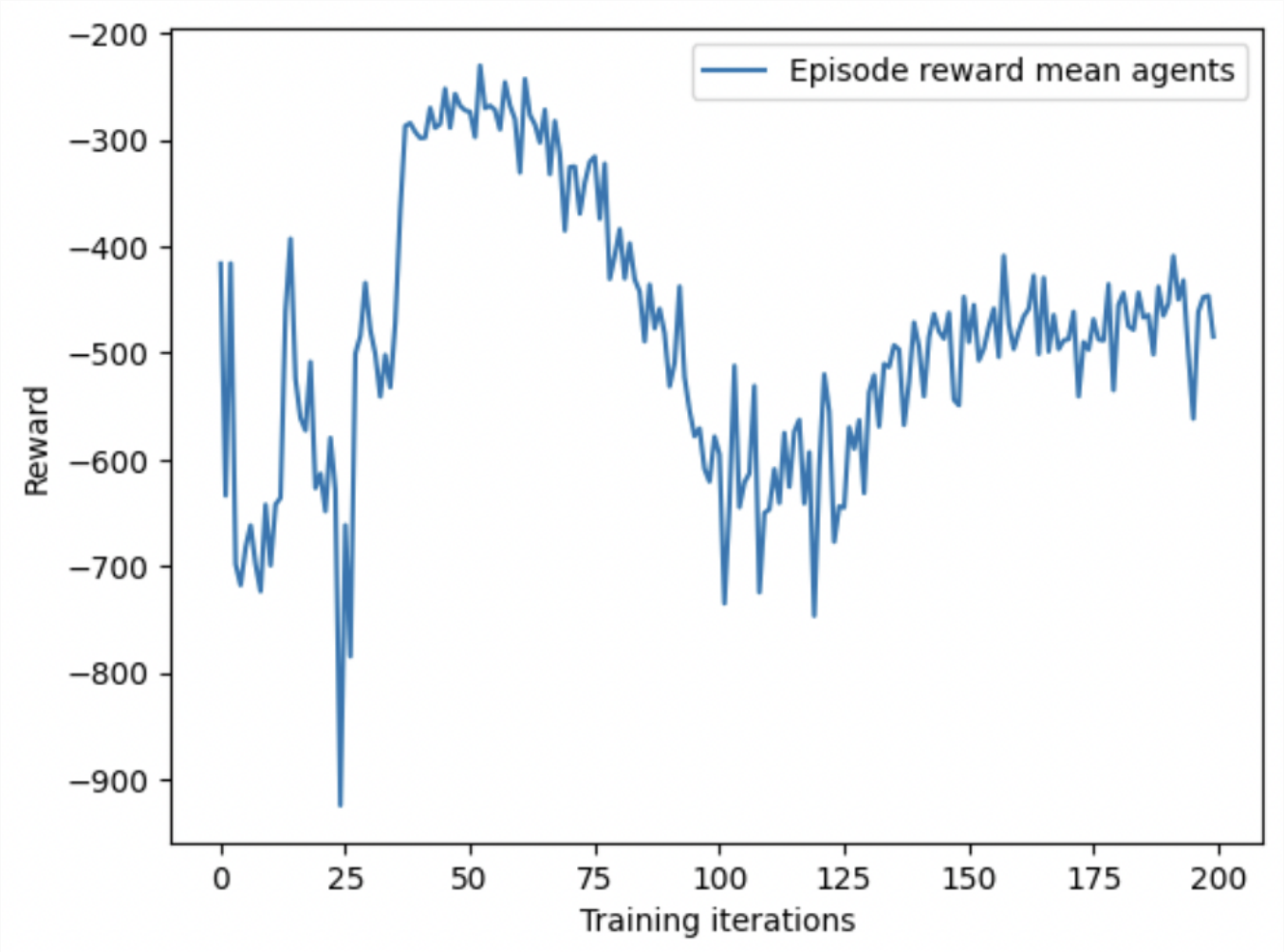
Cooperative Environment – Baseline Reward



Cooperative Environment – Moderate CPT

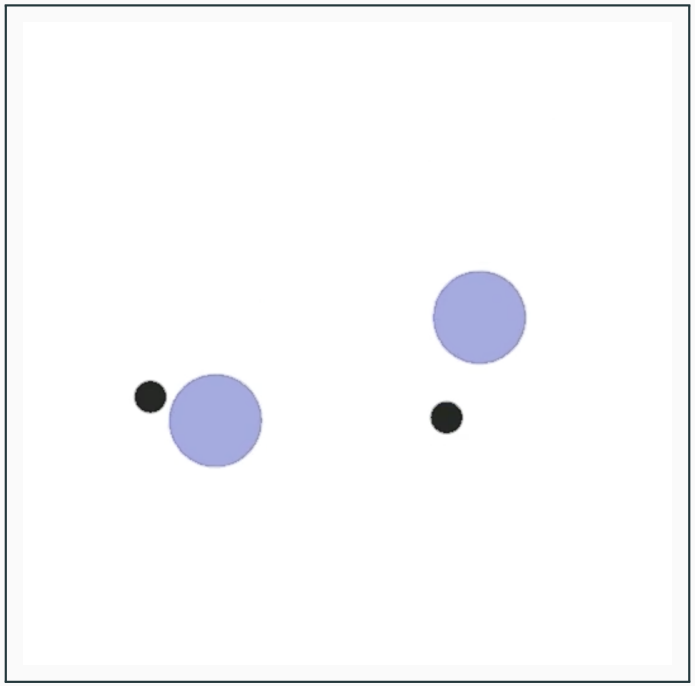


Cooperative Environment – Extreme CPT

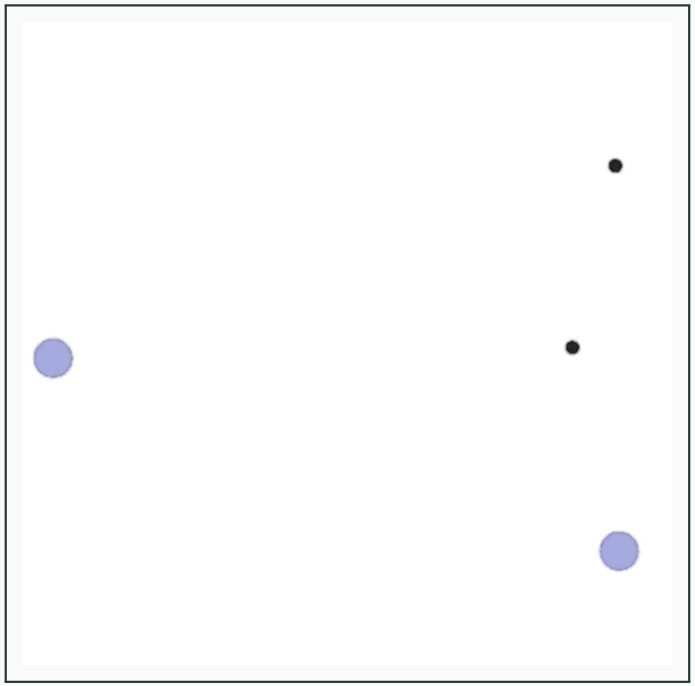


Cooperative Environment - Visualization of MPE

Baseline CPT



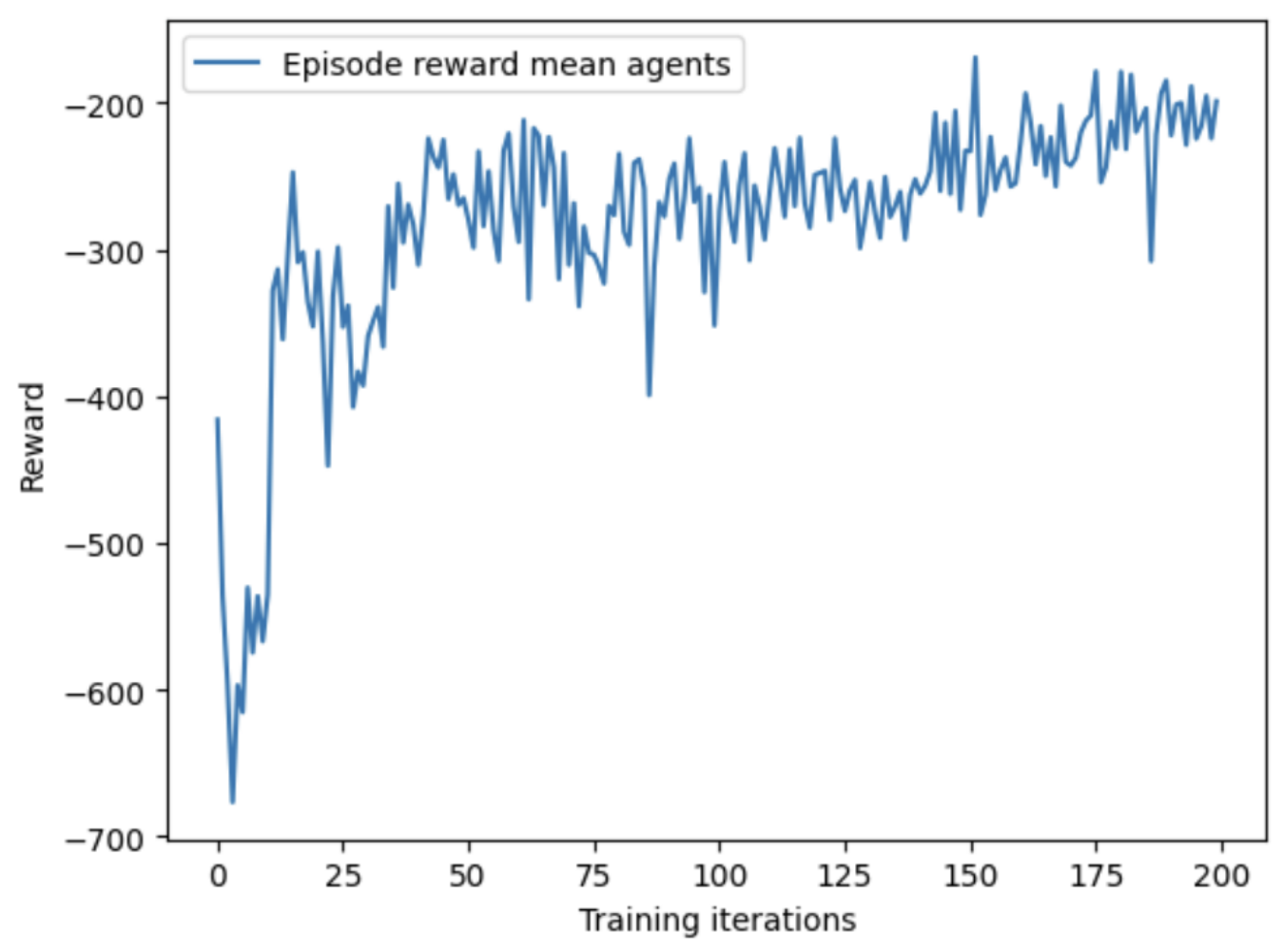
Extreme CPT



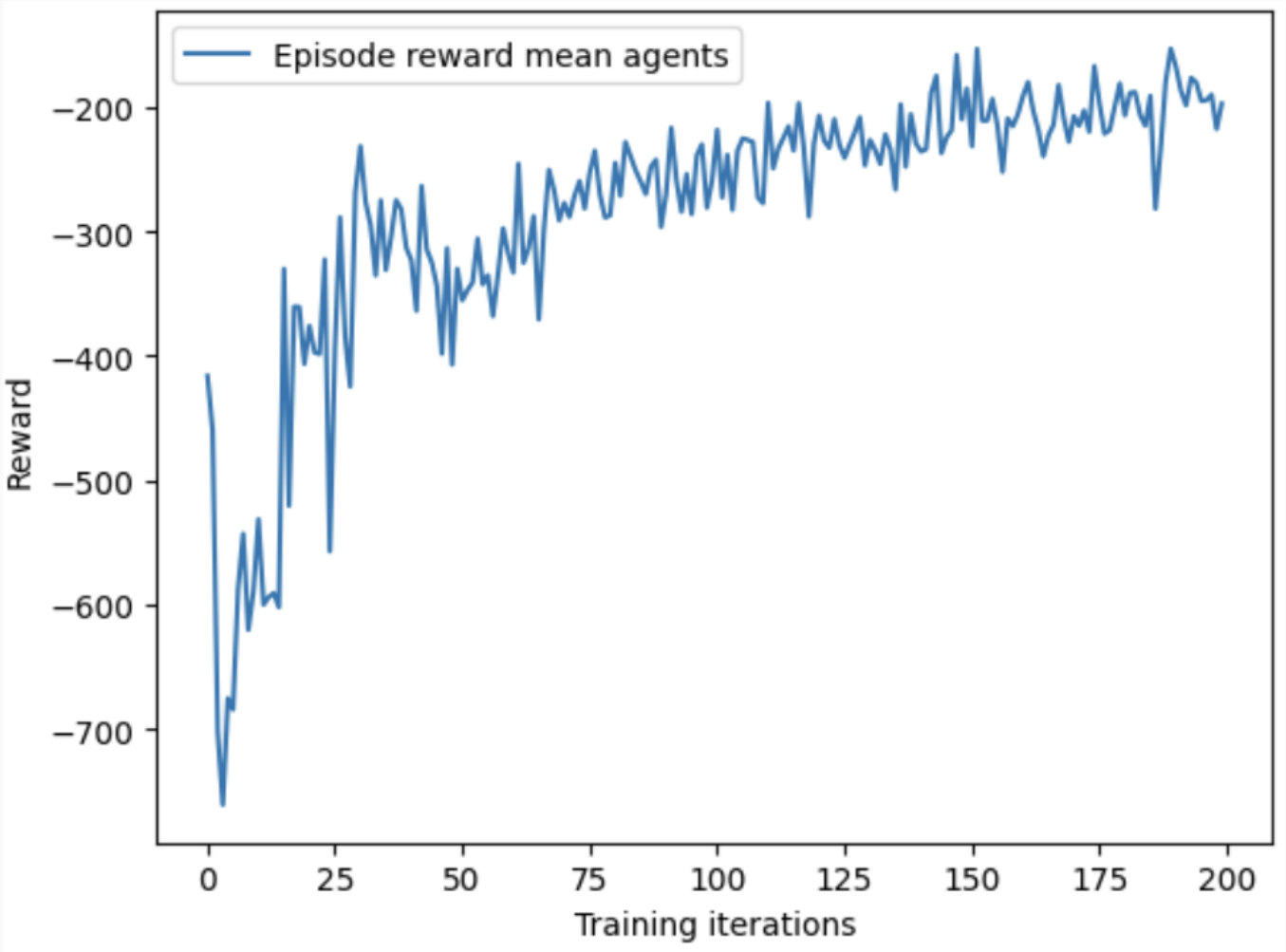
What happens when we allow give two CPT-trained agents access to each others utility and expected reward functions in Simple Spread?

- In all three of the Risk-Averse, Risk-Seeking, and both Risk-Averse and Risk-Seeking, we see convergence to same equilibrium as in base case.
- **What this means?:** When seeing the actions of the other agent, even while being risk averse, the agents still act optimally.

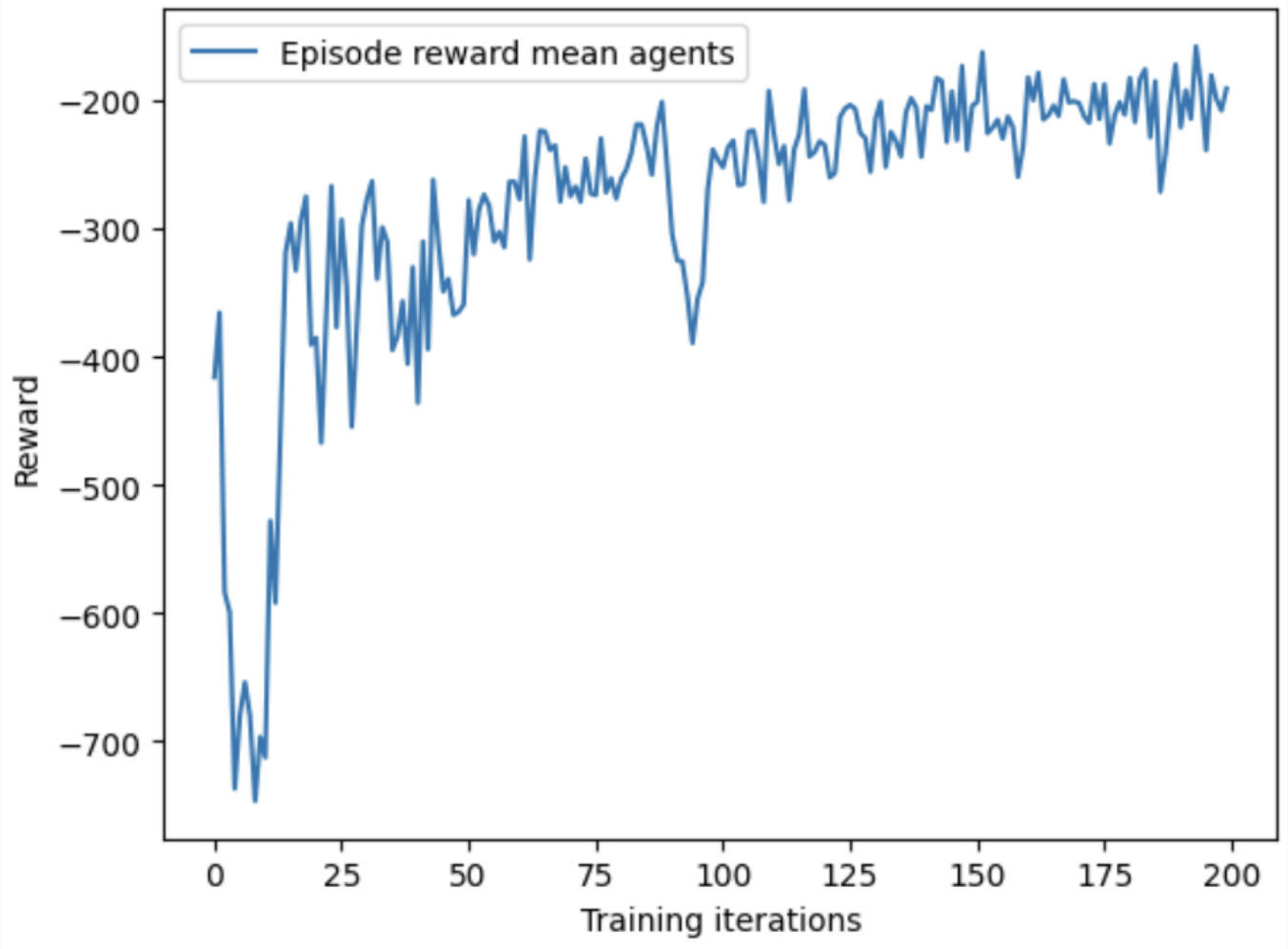
Cooperative Testing – Seeing: Risk Averse



Cooperative Testing – Seeing: Risk Averse & Risk Seeking



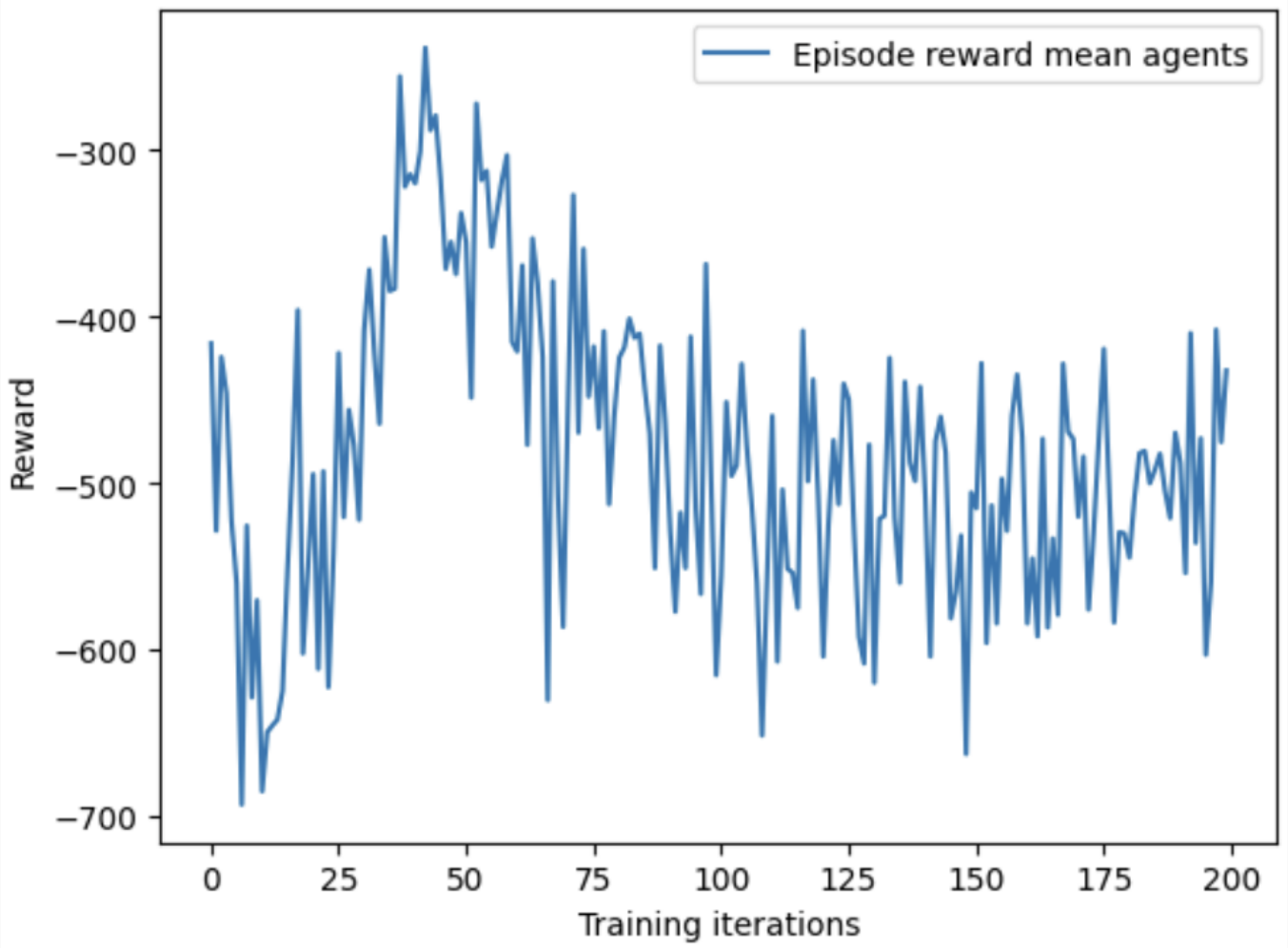
Cooperative Testing – Not Seeing



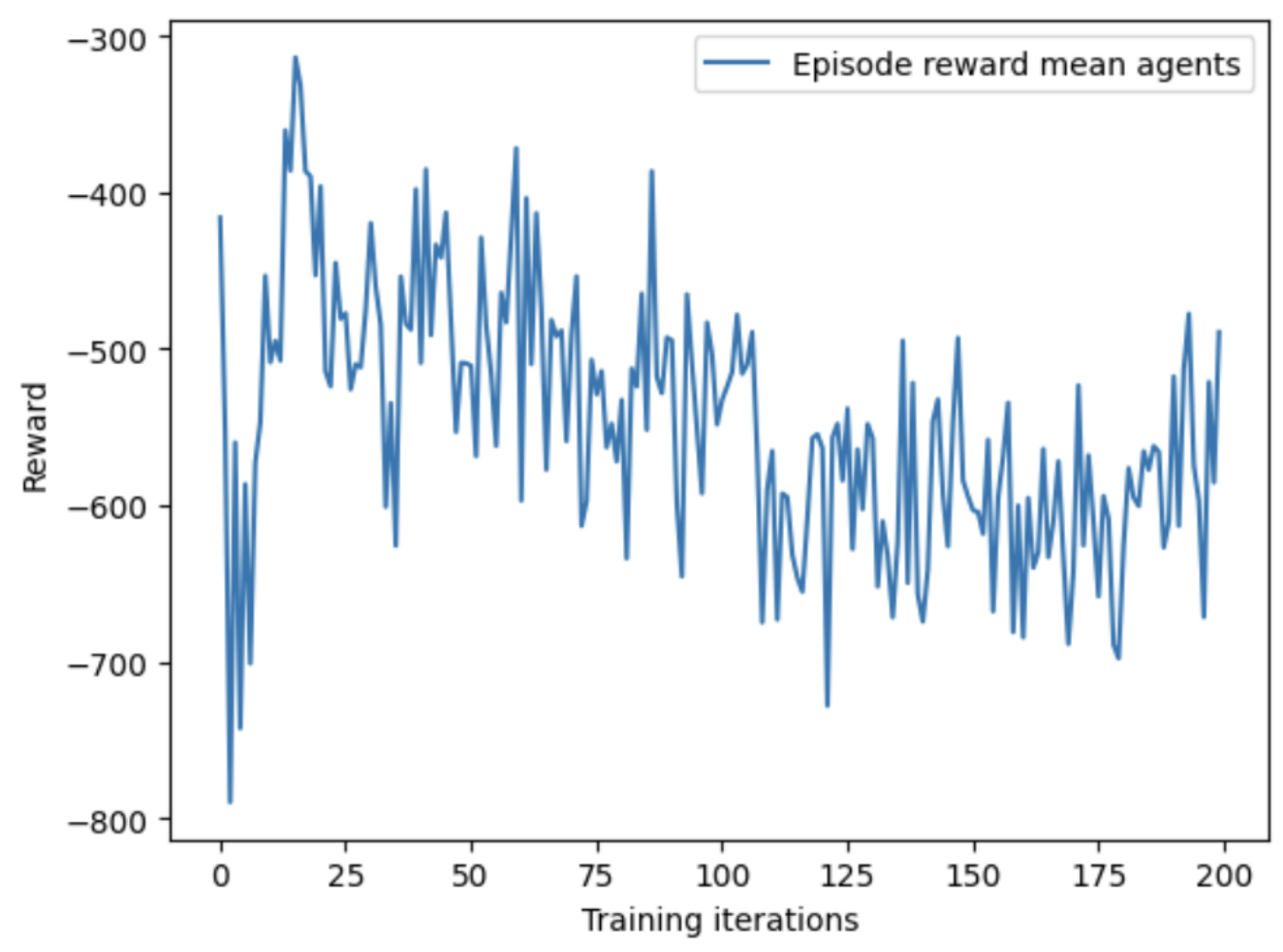
What happens when we give two CPT-trained agents the ability to dynamically choose their CPT-related hyperparameters as they progress through simple spread?

- In general, we see a much **larger swings in the expected reward**. This is true in the risk seeking, as well as moderate and low risk aversion cases.
- In all cases, there is an **unstable equilibrium which is significantly worse** than that of the case where the hyperparameter are consistent.
- Evidence points to inability to properly optimize actions when assuming that risk profile changes very quickly (only after a few iterations).

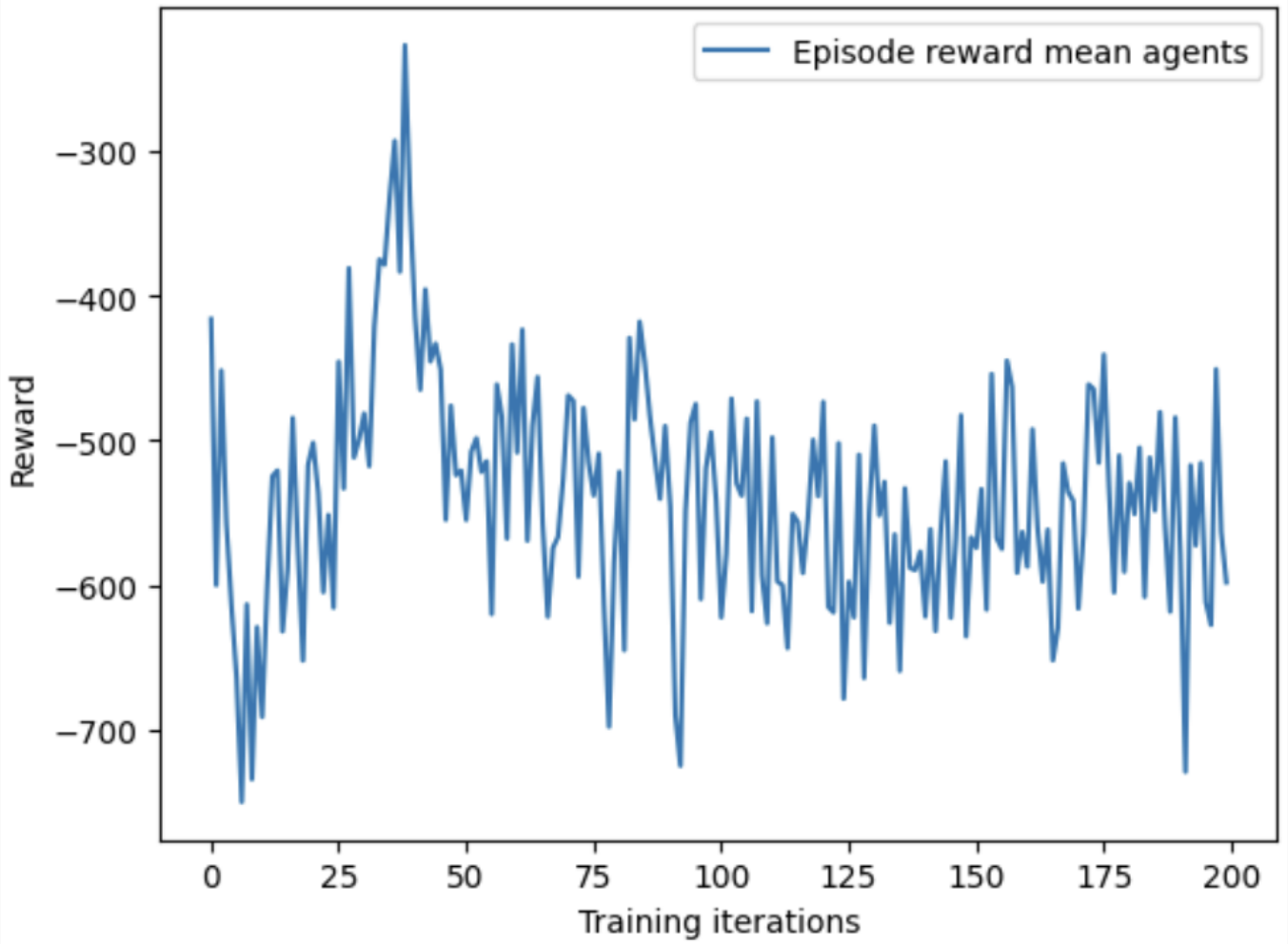
Cooperative Dynamic – Risk Seeking



Cooperative Dynamic – Moderate CPT



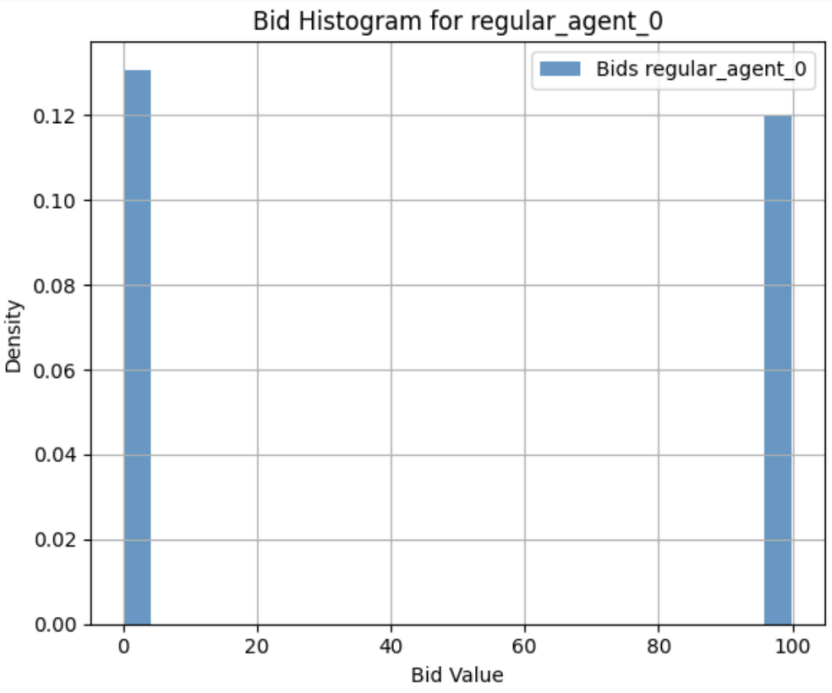
Cooperative Dynamic – Low CPT

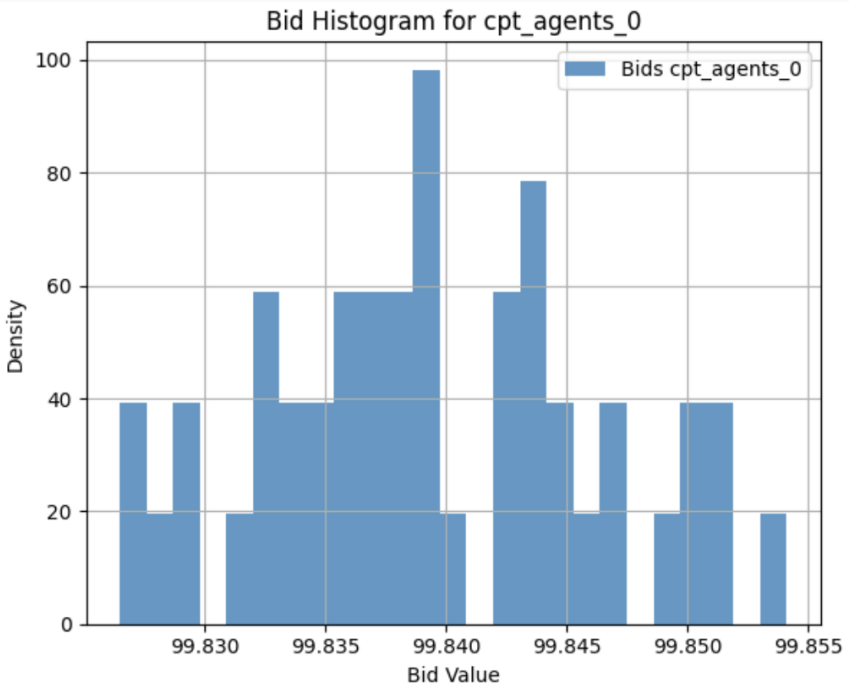


What happens when we put three CPT-trained and three non-CPT agents head to head in a first price auction?

- Behavioral economic theory suggests that CPT-trained agents should be overbidding consistently and as a result, should win most auctions due to risk aversion. This is indeed what we see.
- Non-CPT agents optimize differently, they choose to bet nothing, they minimize their expected losses rather than trying to avoid them. This behavior can likely not be changed in the context of a first price auction.
- Evidence points to the fact that the CPT-training algorithm is making the agents act like humans, using nearly all their available budget in an attempt to win the auction.

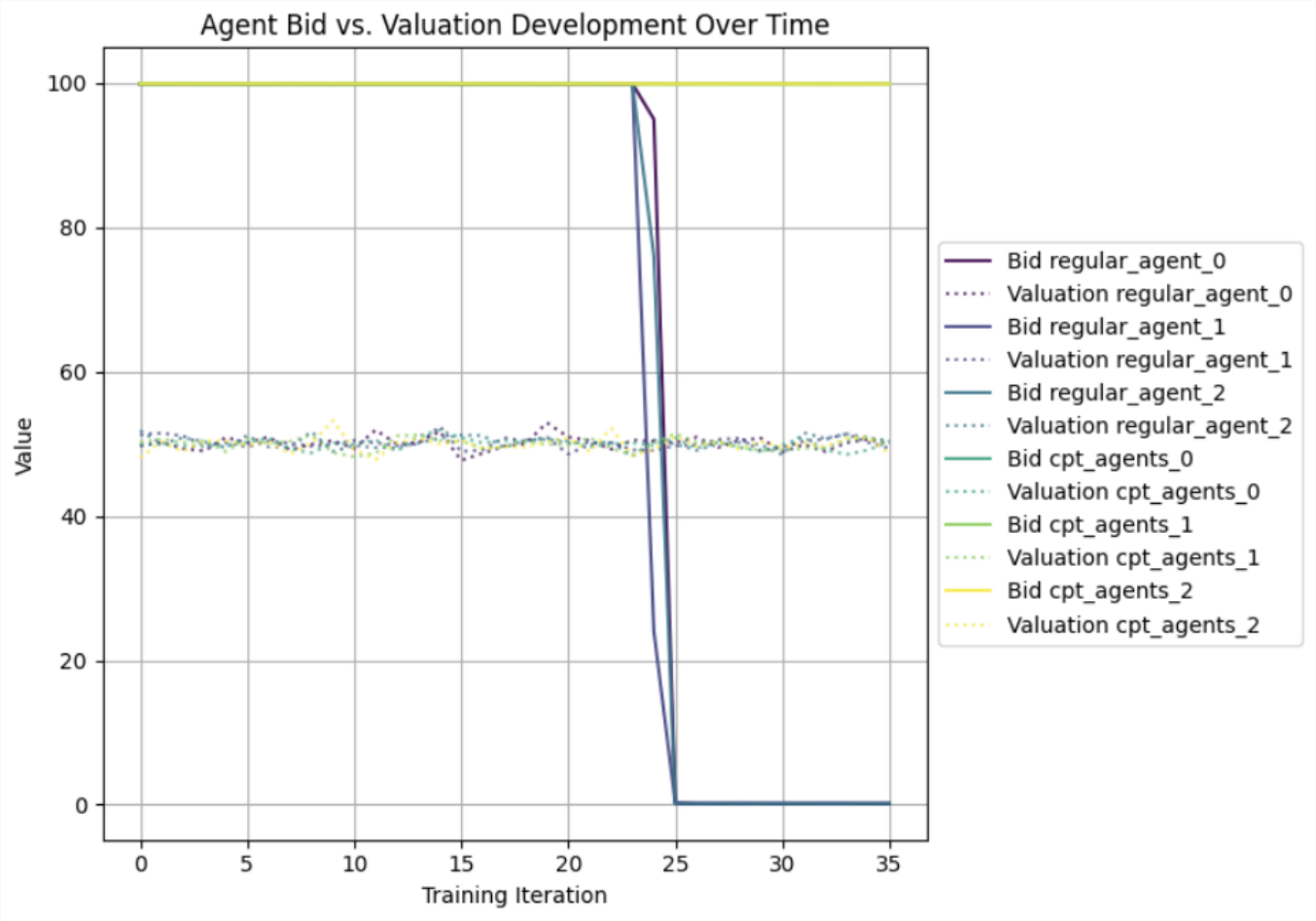
First Price Auction (3 CPT vs. 3 Non-CPT) – Bid Distribution



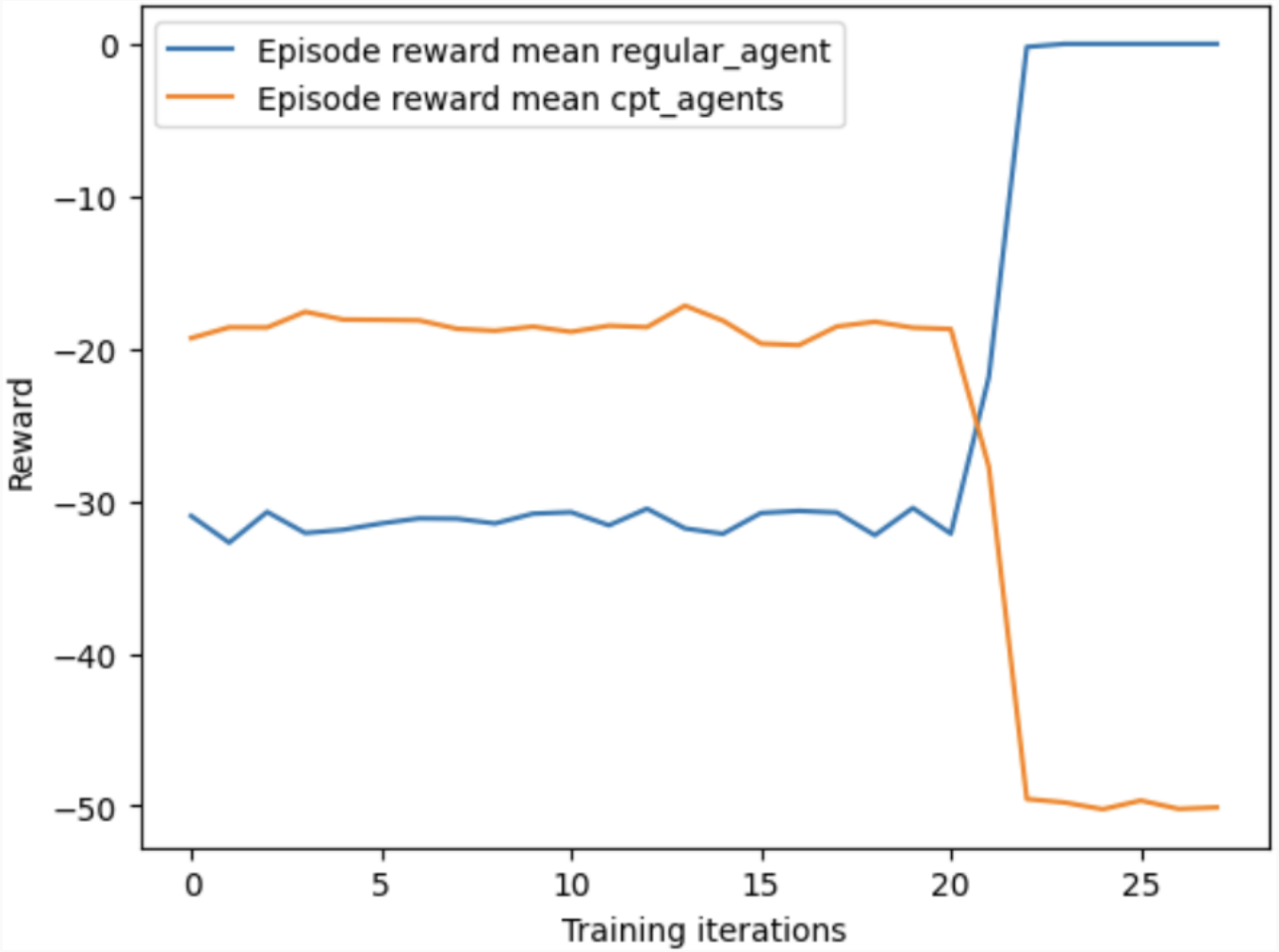


First Price Auction (3 CPT vs. 3 Non-CPT) – Valuation over Iterations

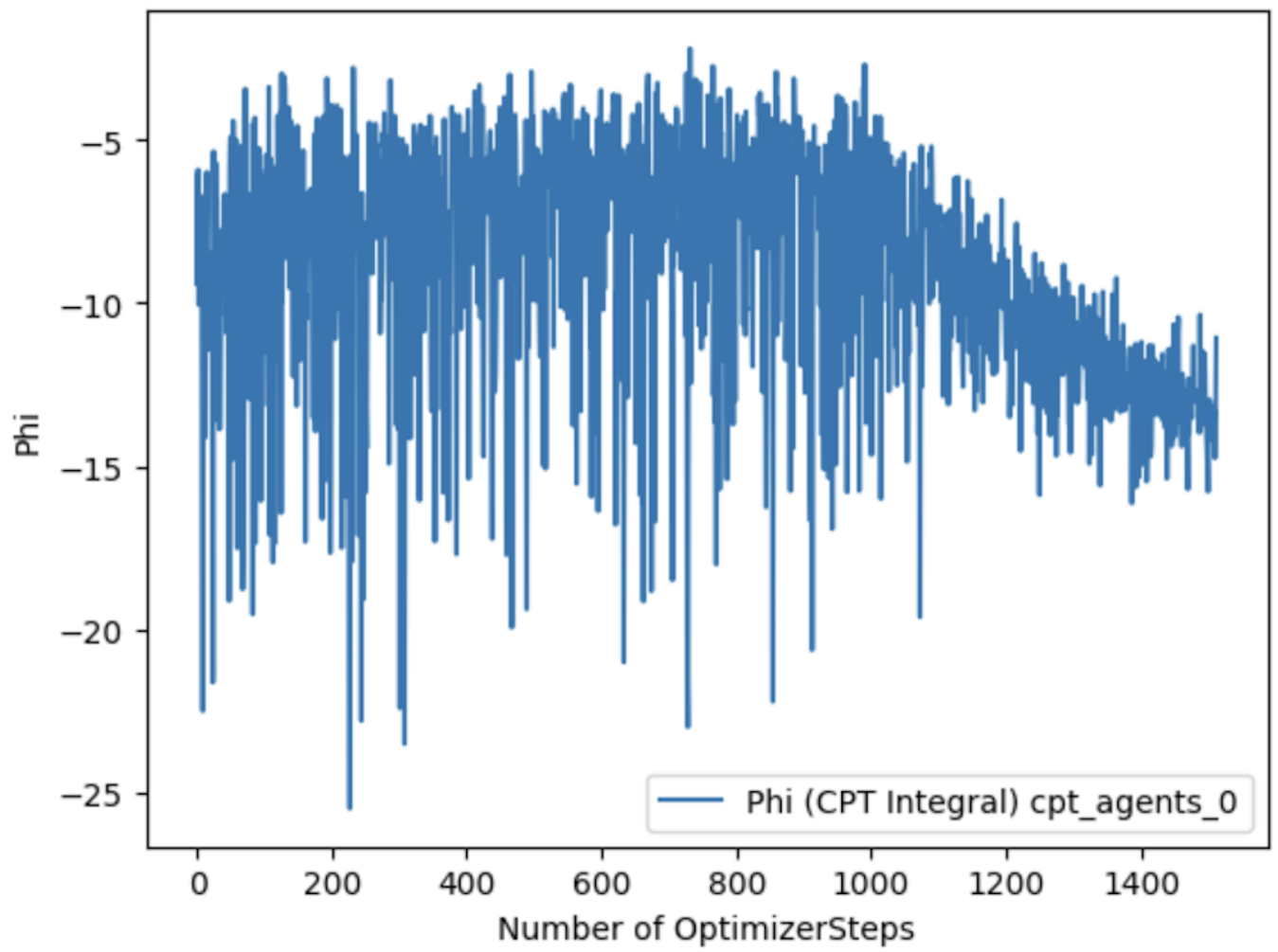
Bidding vs. Valuation – 3 of Each



First Price Auction - Learning



First Price Auction - Integral Trends



Project Reflection

Technically/Conceptually Difficult Challenges

- **CPT Integration Complexity:** Implementing CPT reward transformations and probability weighting accurately within the existing MARL framework (MADDPG) required careful modification of loss functions and gradient calculations. Their was no easy way to do this out of the box.
- **Gradient Stability:** The non-linearity introduced by CPT's value and weighting functions led to potential instabilities in policy gradient updates, necessitating careful tuning and potentially advanced optimization techniques.
- **Nonconvex Optimization:** The CPT-adjusted objective function is inherently nonconvex, making convergence less guaranteed and potentially slower compared to standard RL objectives. We had to determine a computational trade off with performance
- **Hyperparameter Tuning:** Finding the optimal balance between standard MARL hyperparameters (learning rates, exploration noise) and CPT-specific parameters (α, λ, β, weighting function parameters) proved complex and crucial for achieving desired behaviors without destabilizing training.

Project Goals Evolving

Initial: **Bridge standard MARL and human-like decision making**

Project Goals Evolving

Initial: **Bridge standard MARL and human-like decision making**

Stage 1: **Integrate CPT-based reward transformations** in MADDPG for proof-of-concept in MPE and auction tasks

Project Goals Evolving

Initial: **Bridge standard MARL and human-like decision making**

↓

Stage 1: **Integrate CPT-based reward transformations** in MADDPG for proof-of-concept in MPE and auction tasks

↓

Stage 2: **Analyze behavioral impact** — quantify loss aversion, probability weighting, risk-sensitive metrics, and training stability

Project Goals Evolving

Initial: **Bridge standard MARL and human-like decision making**

↓

Stage 1: **Integrate CPT-based reward transformations** in MADDPG for proof-of-concept in MPE and auction tasks

↓

Stage 2: **Analyze behavioral impact** — quantify loss aversion, probability weighting, risk-sensitive metrics, and training stability

↓

Final: **Generalize and align** extend CPT-MARL to large-scale domains (LLM alignment) with interpretable, human-aligned risk preferences

Use of AI Tools

1. **Coding:** Helped provide some help in some of the more complex coding (creating a custom MARL environment).
2. **Explanation of Errors:** There were a lot of times that we ran into errors in torchRL which required us to fundamentally change the way we were framing the problem.
3. **Logical Consistency:** Ensuring the conclusions we were making were logically consistent with the previous research we had been reading.
4. **Reading and Writing:** Before reading any paper on CPT and MARL, we would run the paper through an LLM to see if it was going to provide any useful information in general, especially when we were looking for some specific information.

Personal Contributions

Sheyan

1. Dynamic Hyperparameter selection for the CPT trained agents resulting in more unstable rewards and and no clear stable equilibrium
2. Adaptive Optimization Models (Adam)
3. A more detailed insight into the way that optimization algorithms work and more importantly, why they work
4. Try seeing the effect of our policy on a pretrained LLM for a specific action (Ex: LLM trained to choose surgery cases to take on)
5. To move directly to some complex cases in which CPT observed outcomes would have been very cool (build a very complex and detailed auction or economic environment)

Alok

1. It actually worked on the first few attempts, which is rare lol
2. Replacing nasty things with nicer approximations – however crude, linearization is surprisingly powerful
3. Do not trust, models, figures, parameters, seeds, etc. unless you run it yourself on your computer
4. Can we make informed design choices about large multi-agent systems using small proxy ones: This would suggest that Deep RL training reliably encode CPT dynamics in some scale invariant manner
5. Would have started sooner with a different, more relevant environment + extracted some mathematical guarantees on under what circumstances this stuff works/scales to

Khush

1. The auction setting aligned with the original study
2. "A Playbook for Tuning Deep Learning Models"
3. Practical Experience in implementing complex algorithm
4. Wrap CPT/MADDPG into TRL to align/post-train LLMs
5. Not focusing too much on the crude approximation and getting bogged down in the math, and directly going towards the piecewise approximation

Questions?