Nishant Kumar (grad)

AdX Research Summary

1 Project Overview

The goal of our project is to apply RL to (small instances of) the AdX game (or more generally, sequential auction style games). The particular AdX game variant studied is the **1-campaign, N-day** variant.

The AdX game is interesting to tackle from an RL perspective because it poses several interesting properties and challenges:

- Stochasticity in the game can be a consequence of both the randomness of an impression opportunity's demographic(s) and the randomness of each player's strategy (i.e. players playing mixed strategies).
- Determining an optimal policy via *planning* is difficult because determining a model *a priori* is difficult or infeasible [1].
- The domain offers continuous control, as bids are real-valued. Furthermore, the domain can be highly dimensional, as there can be many types of demographics. Consequently exhaustive search of the space is often infeasible or intractable, thus smart exploration and/or generalization is required.
- The domain can be explored from both a single-agent perspective and a multi-agent perspective.

2 Methodology

We studied single-agent scenarios (i.e. out of M players, only one has an RL policy whereas others have fixed policies) and we focused on policy gradient methods. In particular, the policy is represented by a neural network, which learns a Gaussian distribution using the architecture portrayed in `Figure 1`. Note that the most canonical activation function to use for each layer would be ReLU.

[1] Players are not given the distribution from which impression opportunities are drawn. Furthermore, in game variants in which players are procured campaigns at random, the players are also not given the distribution from which campaigns are drawn.

AdX Research Summary

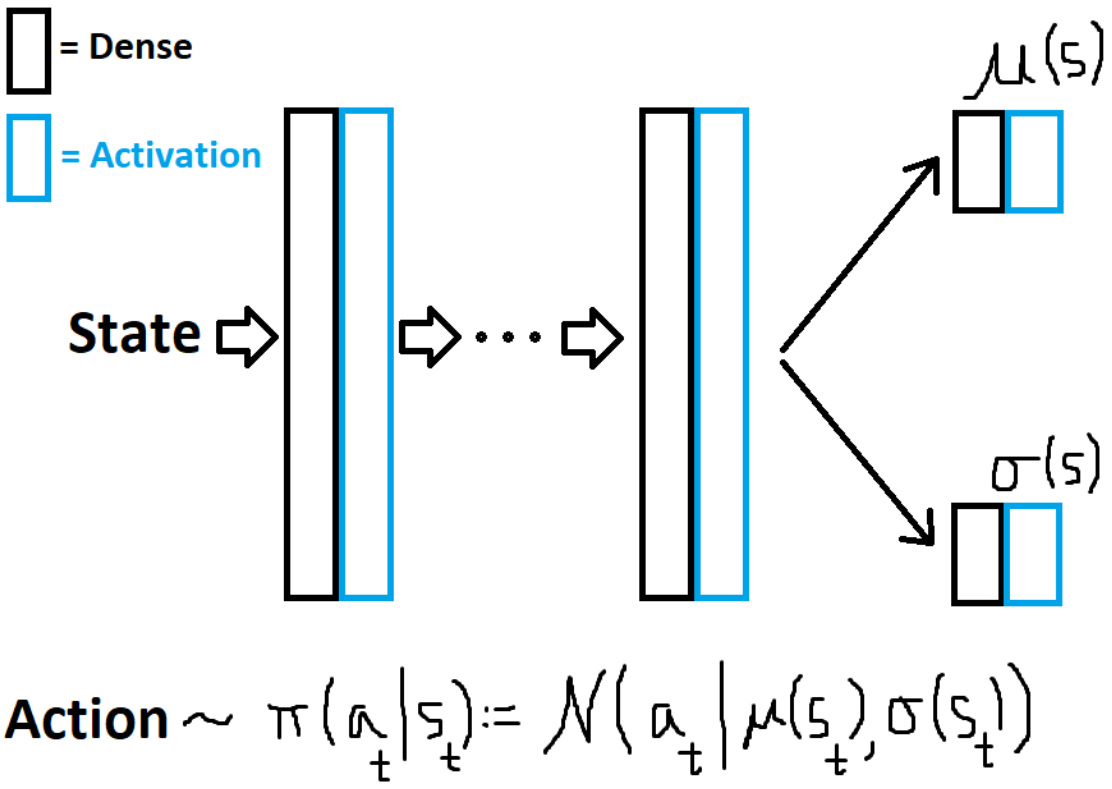

Figure 1: Policy Network Architecture.

This paper will focus on the following algorithms:

- REINFORCE
- REINFORCE with baseline

In the following experimental setups:

- Setup 1
  - The campaign set C ={{reach: 10, budget: \$100, target: {Female}}}. Each $c \in C$ has equal chance of being drawn.
  - The demographic set (for impression opportunities) $D = \{\{\text{Female}\}\}$. Each $d \in D$ has equal chance of being drawn.
  - $k \in [1, 2)$ impression opportunities are created each day of the game.
  - There are two players in the system: one RL agent and one fixed strategy player. The fixed strategy is `FixedBidPolicy(5, 5)`: every day, the player bids (on its campaign's target) \$5 per impression, up to \$5 total that day.

- Setup 3
  - C ={
    {reach: 10, budget: \$100, target: {Male}},
    {reach: 10, budget: \$100, target: {Female}} }.
    Each $c \in C$ has equal chance of being drawn.
  - D = {{Male}, {Female}}. Each $d \in D$ has equal chance of being drawn.
  - $k \in [1, 4)$ impression opportunities are created each day of the game.
  - There are two players in the system: one RL agent and one fixed strategy player. The fixed strategy is `FixedBidAdaptiveLimitPolicy(5)`: every day, the player bids (on its campaign's target) \$5 per impression, up to $\$5 \times (\text{reach} - \text{impressions})$ total that day.
- Setup 4
  - C ={
    {reach: 10, budget: \$100, target: {Male}},
    {reach: 10, budget: \$100, target: {Female}} }.
    Each $c \in C$ has equal chance of being drawn.
  - D = {{Male}, {Female}}. Each $d \in D$ has equal chance of being drawn.
  - $k \in [10, 21)$ impression opportunities are created each day of the game.
  - There are two players in the system: one RL agent and one fixed strategy player. The fixed strategy is `FixedBidAdaptiveLimitPolicy(5)`.
- Setup 5
  - C ={
    {reach: 10, budget: \$100, target: {Male}},
    {reach: 10, budget: \$100, target: {Female}},
    {reach: 10, budget: \$200, target: {Young}},
    {reach: 10, budget: \$150, target: {Old}} }.
    Each $c \in C$ has equal chance of being drawn.
  - D = {{Male}, {Female}, {Young}, {Old}}. Each $d \in D$ has equal chance of being drawn.
  - $k \in [10, 21)$ impression opportunities are created each day of the game.
  - There are two players in the system: one RL agent and one fixed strategy player. The fixed strategy is `FixedBidAdaptiveLimitPolicy(5)`.
- Setup 6

- C ={
  {reach: 10, budget: \$100, target: {Male}},
  {reach: 10, budget: \$100, target: {Female}},
  {reach: 20, budget: \$200, target: {Male}},
  {reach: 20, budget: \$200, target: {Female}},
  {reach: 30, budget: \$250, target: {Male}},
  {reach: 30, budget: \$250, target: {Female}} }.
  The weights (i.e. probs) of each $c \in C$ being drawn (in corresponding order): $\{3, 3, 2, 2, 1, 1\}$.
- $D = \{\{\text{Male}\}, \{\text{Female}\}\}$. Each $d \in D$ has equal chance of being drawn.
- $k \in [1, 3)$ impression opportunities are created each day of the game.
- There are two players in the system: one RL agent and one fixed strategy player. The fixed strategy is `FixedBidAdaptiveLimitPolicy(5)`.

[NOTE: all setups use a first-price auction.]

[NOTE: for all versions of all algorithms we do state and action scaling. This is important so as to keep the network weights from blowing up.]

3 Hypothesis

Applying "out-of-the-box" REINFORCE will result in poor-to-mediocre performance in all setups. Tuning in various ways (to better account for the domain) can result in noticeable improvements. "Out-of-the-box" means using canonical or textbook implementations, e.g. a policy neural network with randomly initialized weights and all ReLU activation functions, fixed learning rate, Gaussian distribution, etc. Although tuning along any single axis gives only minor improvements (e.g. 1% - 5%), stacking tuning across multiple axes gives noticeable improvements.

4 Results

Some context:

- **game_length** represents how many days each game instance lasts.
- **num_trajs** represents the number of (independent) game instances (i.e. batch size) used for each epoch of training.
- **num_epo** represents the number of iterations of training.
- For visual clarity, std. dev. intervals are not plotted on the graphs.

- REINFORCE_Gaussian_v6_1
  - "Out-of-the-box" algorithm; random weight initialization, ReLU activation on all layers.
- REINFORCE_Gaussian_v6_2
  - Same as v6_1 algorithm, but with softplus activation on μ and σ layers.
- REINFORCE_Gaussian_v2
  - Tuned algorithm
  - Weights initialized so that Gaussian is centered around budget/reach. Furthermore, policy network learns an offset from this point.
  - Leaky ReLU on all layers, but softplus on μ and σ layers.
- REINFORCE_Gaussian_v3
  - Almost the same as v2 - ELU on all layers (instead of leaky ReLU), but softplus on μ and σ layers.
- REINFORCE_Baseline_Gaussian_v2
  - Same as v2, but with learned value function V as a baseline.

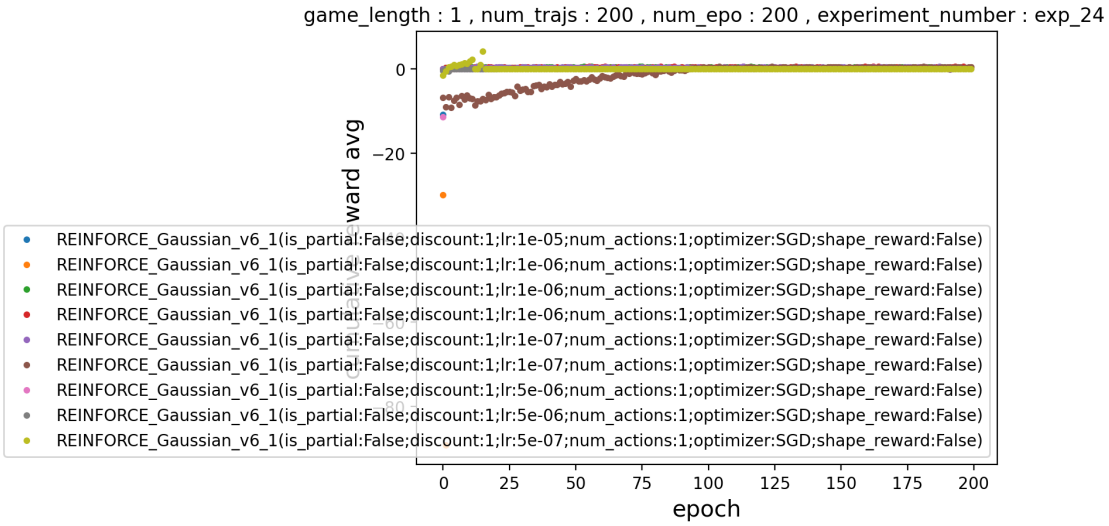

Figure 2: REINFORCE_Gaussian_v6_1

`Fig 2` comments:

- Zero collapses almost immediately, for many learning rates.
- These are the "good" runs; each learning rate plot here took many restarts in order to generate something that either didn't immediately zero collapse or immediately blow up. In other words, it was quite difficult to generate "good" results.
- Results were so poor that I used v6_2 for the rest of the comparisons below.

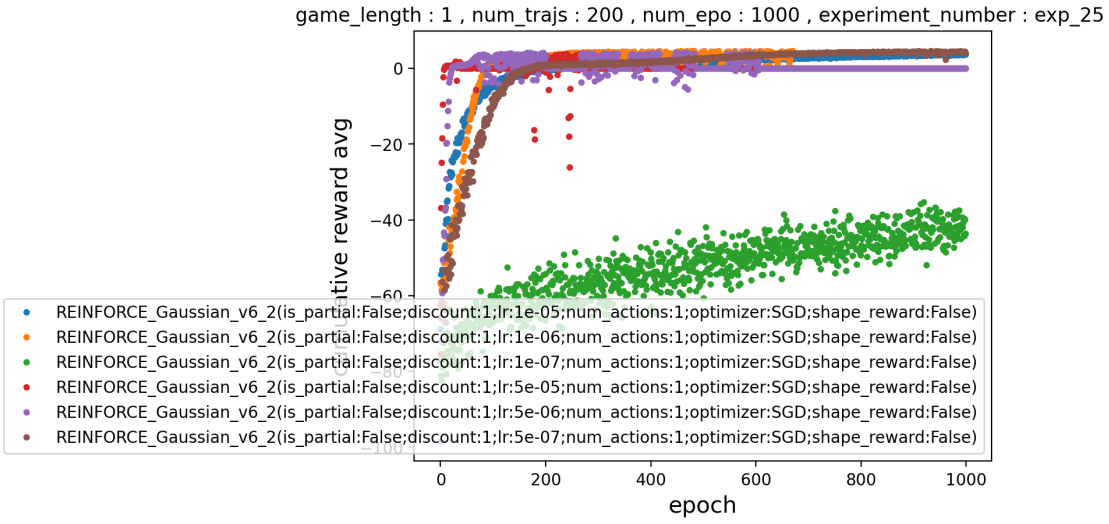

Figure 3: REINFORCE_Gaussian_v6_2

`Fig 3` comments:

- Does better than v6_1, but still not great.
- Zero collapses for some learning rates, and still doesn't perform very well in non-zero-collapse cases.

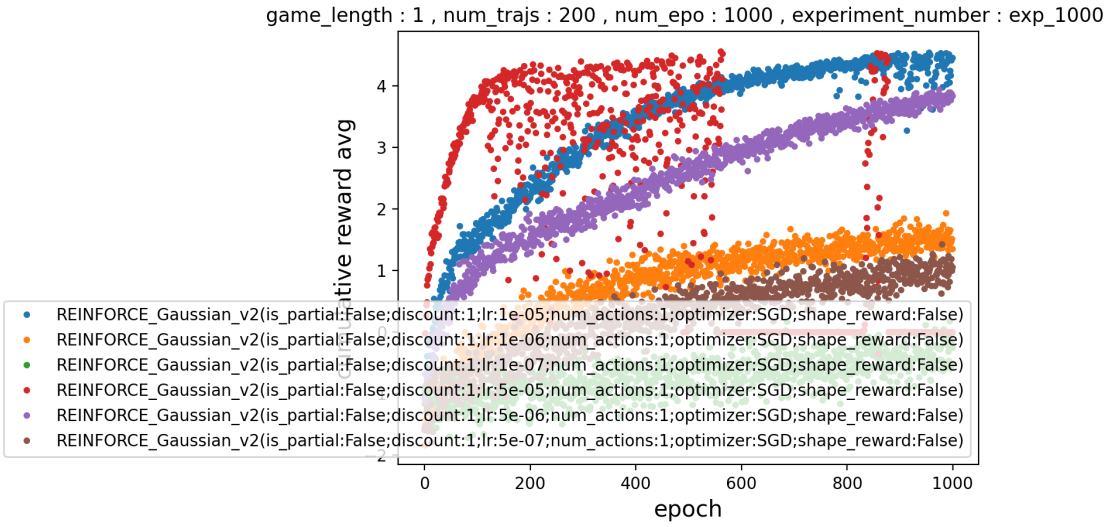

Figure 4: REINFORCE_Gaussian_v2

`Fig 4` comments:

- Shows (roughly) monotonic improvement for most learning rates.
- Behaves as one would expect for a gradient-based algorithm - gets better as learning rate increases, until it's increased too much.

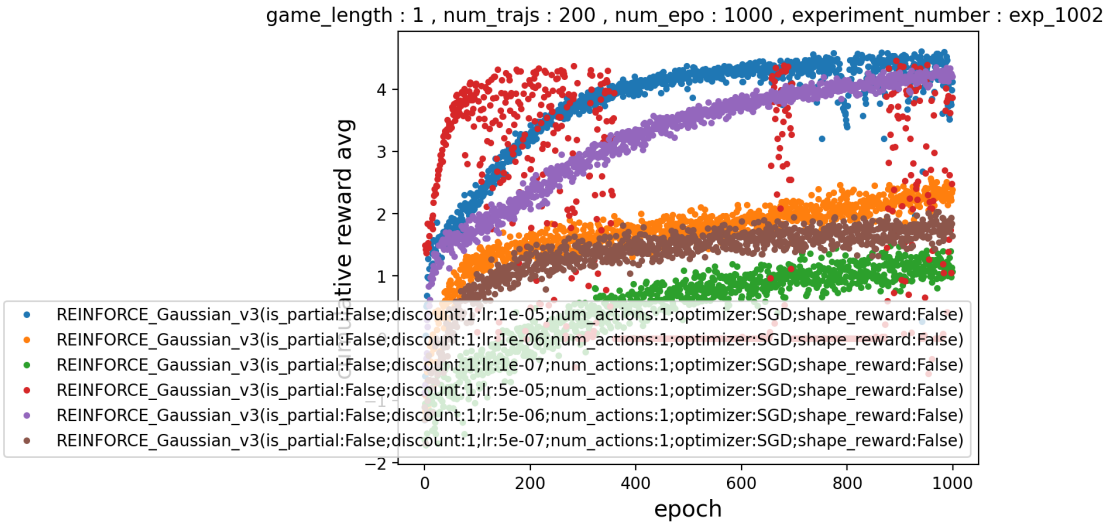

Figure 5: REINFORCE_Gaussian_v3

`Fig 5` comments:

- Shows good performance, like v2.
- Notable differences: rises faster initially, but might "saturate" sooner than v2.

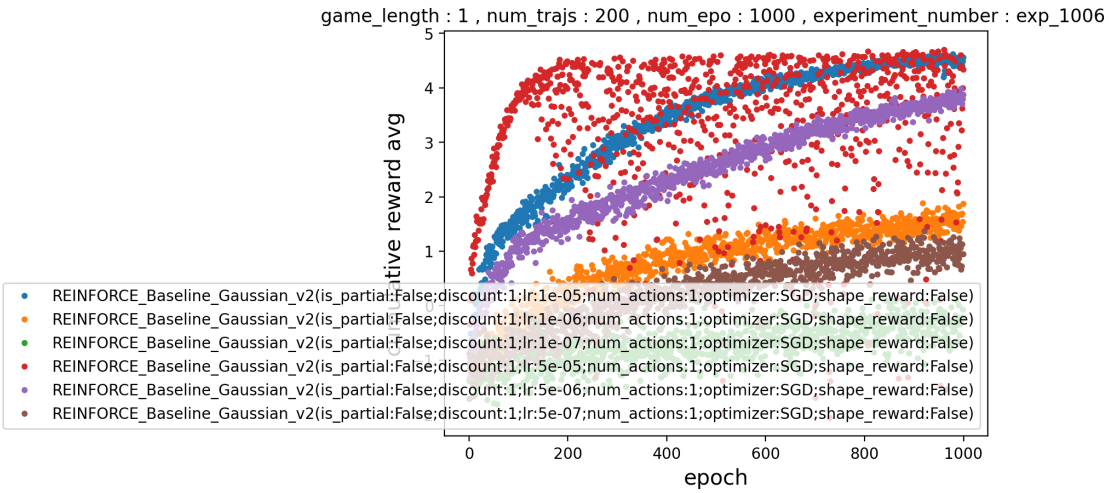

Figure 6: REINFORCE_Baseline_Gaussian_v2

`Fig 6` comments:

- Shows good performance, like regular v2.

- Seems to have lower variance and be more stable (though hard to tell in 1000 epochs). See next figure for a better comparison.

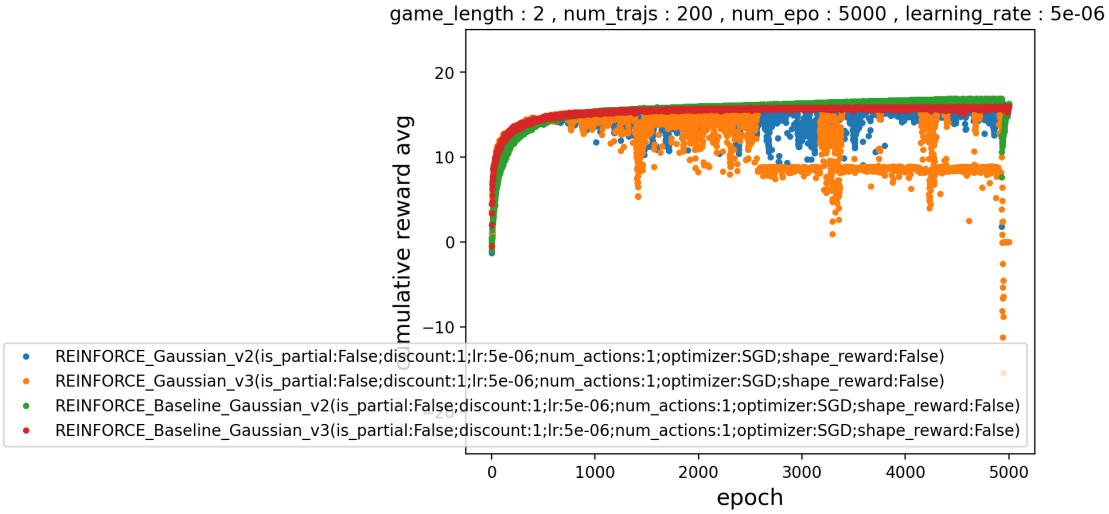

Figure 7: no baseline vs. baseline

`Fig 7` comments:

- Better diagram to compare baseline performance; non-baseline versions experience higher variance and stability issues.

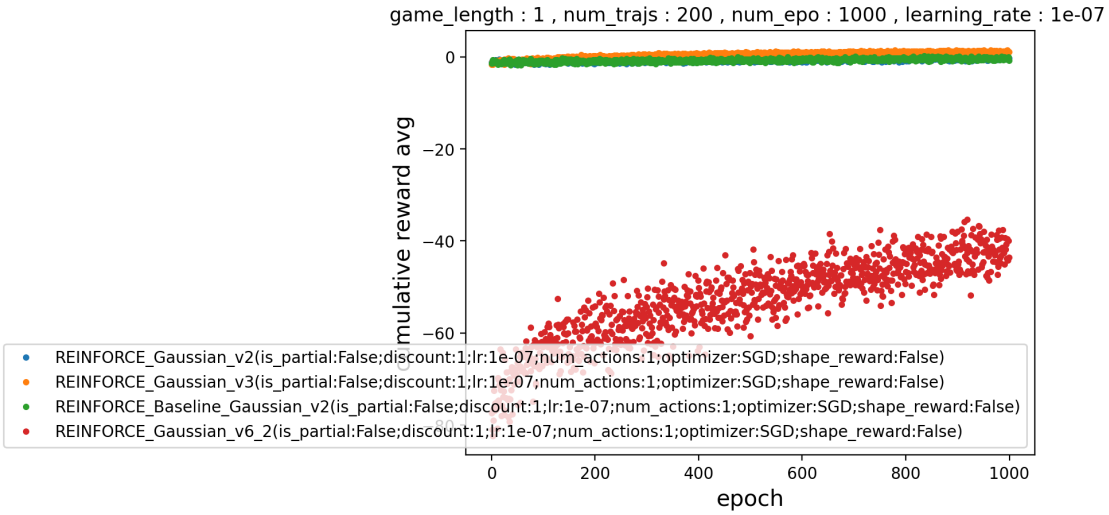

Figure 8: learning rate: 1e-7

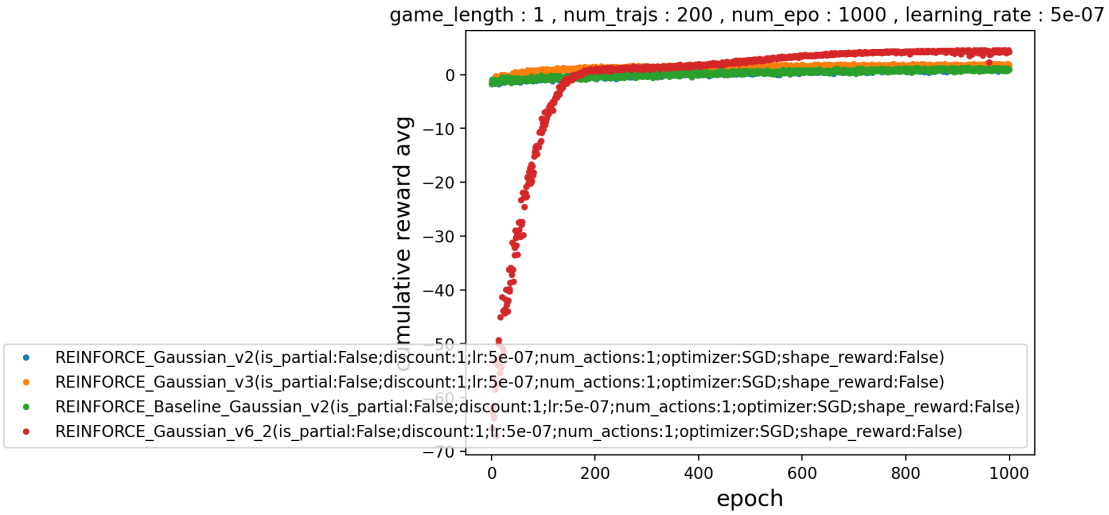

Figure 9: learning rate: 5e-7

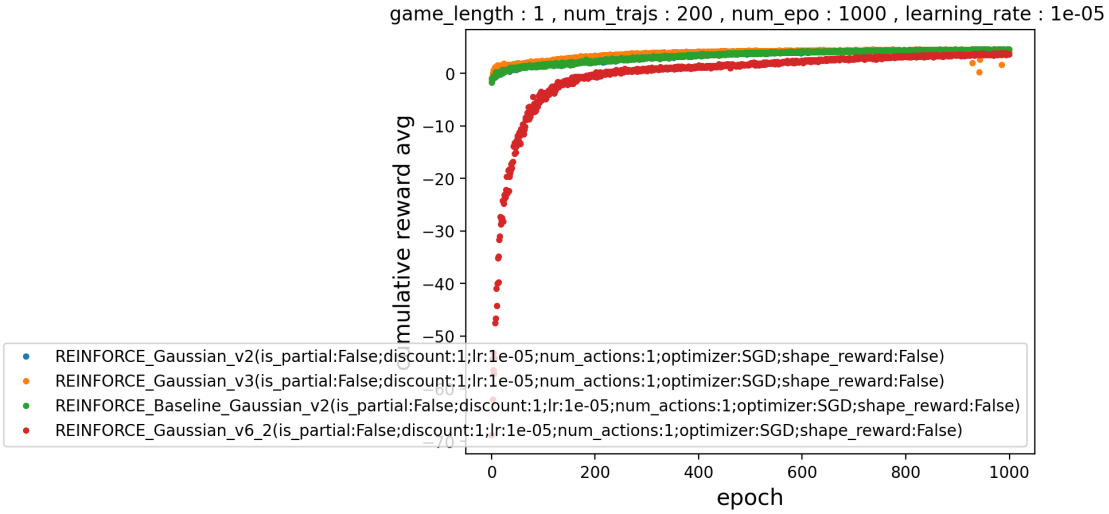

Figure 10: learning rate: 1e-5

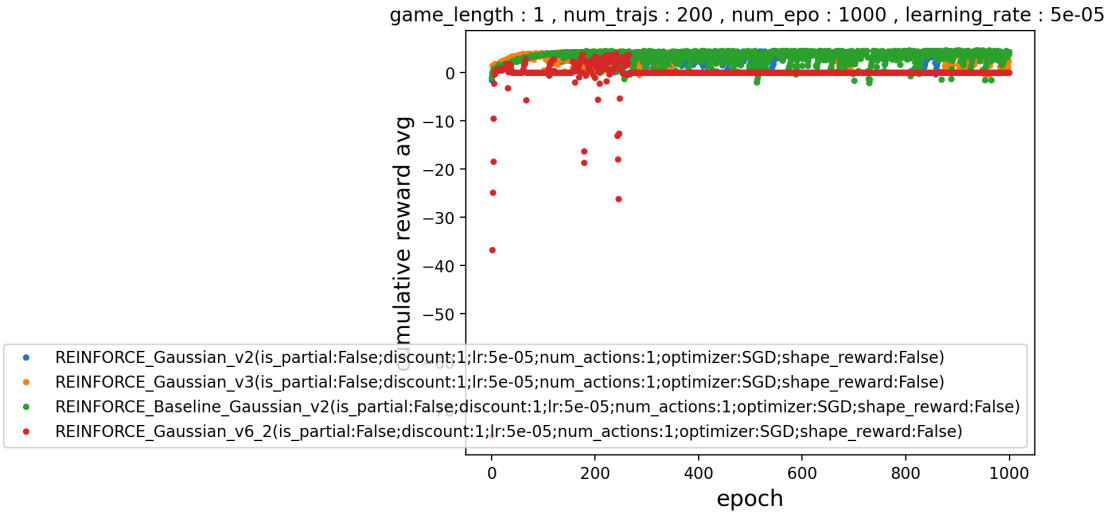

Figure 11: learning rate: 5e-5

TODO: add other setup diagrams here.

5 Appendix

5.1 Formal RL Definition

5.1.1 Variable Definitions

- Let m be the number of agents in the system.
- Let $a_t^{(i)}$ represent the action agent i takes at time t, for $i \in [1, \ldots, m]$.
- Let $s_t \in S$ represent a full, joint state for the entire system (i.e. all agents' states), at each time step t.
- Let $r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)})$ be the reward function for agent i.
- Let π_{θ_i} be the policy for agent i.
- Let $p_\theta(\tau)$ be the probability of seeing joint trajectory τ:

$$p_\theta(\tau) := p_\theta(s_0, a_0^{(1)}, \ldots, a_0^{(m)}, s_1, a_1^{(1)}, \ldots, a_1^{(m)}, \ldots, s_{T-1})$$

  and more generally:

$$p_\theta(\tau_t) := p_\theta(s_t, a_t^{(1)}, \ldots, a_t^{(m)}, s_{t+1}, a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)}, \ldots, s_{T-1})$$

- Let $J(\theta_i)$ be the (undiscounted) objective function for agent i:

$$J(\theta_i) := \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) \right] := \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[r^{(i)}(\tau) \right]$$

- Let value functions $Q^{\pi_{\theta_i}}$ and $V^{\pi_{\theta_i}}$ for agent i with policy π_{θ_i} be defined as follows:

$$V^{\pi_{\theta_i}}(s_t) := \mathbb{E}_{\tau \sim p_\theta(\tau_t)} \left[\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \ldots, a_{t'}^{(m)}) \middle| s_t \right]$$

$$Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) := \mathbb{E}_{\tau \sim p_\theta(\tau_t)} \left[\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \ldots, a_{t'}^{(m)}) \middle| s_t, a_t^{(1)}, \ldots, a_t^{(m)} \right]$$

5.1.2 Assumptions

- Assume users (i.e. impression opportunities) are exogenous.
- Assume campaign distributions are independent.

- Assume all instances of the game use a pre-determined fixed set of campaigns C.
- Full observability equivalently means "every agent sees other agents' states". Without this assumption, state aliasing can occur.
- Assume agents must take actions independently (cond. indep. based on state) and simultaneously at each time step t (i.e. no pre-determined collusion/coordination). Then:

$$
\begin{aligned}
p_\theta(\tau) &= p_\theta(s_0, a_0^{(1)}, \dots, a_0^{(n)}, s_1, a_1^{(1)}, \dots, a_1^{(m)}, \dots, s_{T-1}) \\
&= p(s_0) \prod_{t=0}^{T-2} \pi_\theta(a_t^{(1)}, \dots, a_t^{(m)} | s_t) p(s_{t+1} | s_t, a_t^{(1)}, \dots, a_t^{(m)}) && \text{(Markov prop.)} \\
&= p(s_0) \prod_{t=0}^{T-2} \pi_\theta(a_t^{(1)}, \dots, a_t^{(m-1)} | a_t^{(m)}, s_t) \pi_{\theta_m}(a_t^{(m)} | s_t) p(s_{t+1} | s_t, a_t^{(1)}, \dots, a_t^{(m)}) \\
& && \text{(product rule)} \\
&= p(s_0) \prod_{t=0}^{T-2} \pi_\theta(a_t^{(1)}, \dots, a_t^{(m-1)} | s_t) \pi_{\theta_m}(a_t^{(m)} | s_t) p(s_{t+1} | s_t, a_t^{(1)}, \dots, a_t^{(m)}) \\
& && \text{(assume cond. indep.)} \\
&\vdots \\
&= p(s_0) \prod_{t=0}^{T-2} \left(\prod_{j=1}^{m} \pi_{\theta_j}(a_t^{(j)} | s_t) \right) p(s_{t+1} | s_t, a_t^{(1)}, \dots, a_t^{(m)}) && \text{(repeat } m \text{ times)}
\end{aligned}
$$

5.1.3 Game Definitions

Markov Game
A fully-observable Markov Game $\langle S, A, R, T, \gamma \rangle$ for this game variant would be:

- Let $S = \prod S^{(i)}$, where state $s^{(i)} \in S^{(i)}$ for agent i would be:
  - **campaign**: a campaign initially assigned at the beginning of the game. The agent should attempt to fulfill the campaign by the end of the game (i.e. after N days). A campaign consists of:
    - **target** (a.k.a market segment): a demographic(s) to be targeted.
      There are 26 market segments in this game, corresponding to combinations of {Male, Female} x {HighIncome, LowIncome} x {Old, Young}. A market segment might target only one of these attributes (for example, only Female) or two (Female_Young) or three (Female_Old_HighIncome).
    - **reach**: the number of impression opportunities to capture in the target market segment.

    - ∗ **budget**: the amount of money the advertiser will pay if the campaign is fulfilled.
  - **spend**: the amount spent so far in fulfilling this campaign [2].
  - **impressions**: the number of (relevant) impressions achieved so far. Note that impressions ≤ reach.
  - **day**: the day of the game (i.e. time t).

- Let $A = \prod A^{(i)}$, where action $a^{(i)} \in A^{(i)}$ for agent i would be:
  - $\{(b_1, u_1), \ldots, (b_{26}, u_{26})\}$
    Where $b_j \in \mathbb{R}$, $u_j \in \mathbb{R}$.
    Where each subaction (b_j, u_j) is the **bid per impression** and **total spend limit** for market segment j.
    Note that this represents a bid bundle.

- Let $r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)})$ for each agent i be:

$$r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) := \text{how much earned on day } t - \text{how much spent on day } t$$

  A couple of subtleties here [3] [4].

- Let T be a probability function that represents the probability of transitioning from state s_t to state s_{t+1} after bids have been placed and the auction has been resolved. Namely:

$$T(s_t, a_t^{(1)}, \ldots, a_t^{(m)}, s_{t+1}) := p_\theta(s_t, a_t^{(1)}, \ldots, a_t^{(m)}, s_{t+1})$$

  Where stochasticity is due to exogenous users (i.e. it is random which impression opportunities are present on a given day).

Markov Decision Process
Let agent x be a learning agent and let all other agents have fixed policies. A fully-observable MDP $\langle S, A, R, T, \gamma \rangle$ w.r.t. agent x would be:

[2] Spend is needed in the state in order to avoid state aliasing

[3] On most days of the game, an agent receives non-positive reward because it spends but doesn't earn (recall payout only happens on the last day). On the last day, the agent participates in its last auction, after which the game ends and payout occurs. Thus only on the last day can an agent receive positive reward.

[4] Technically, $r_t^{(i)} \sim P(r|s_t, a_t^{(1)}, \ldots, a_t^{(m)})$, since the reward each day is based on the impression opportunities sold each day, which in turn are drawn randomly and independently of the state and agent actions. However, it can be shown that $r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}, s_{t+1}) = \mathbb{E}_{r_t^{(i)} \sim P(r|s_t, a_t^{(1)}, \ldots, a_t^{(m)})}\left[r_t^{(i)}\right]$. The proof is excluded from this paper.

- Let $S = S^{(x)} \times S^{(-x)}$, where (x) denotes agent x and $(-x)$ denotes all the other agents. Let state $s^{(i)} \in S^{(i)}$ for agent i be the same as in the MG.
- Let action $a \in A$ for agent x be the same as in the MG.
- Let $r(s_t, a_t) := r^{(x)}(s_t, \pi_{\theta_1}(s_t), \ldots, a_t, \ldots, \pi_{\theta_m}(s_t))$
  where $r^{(x)}$ is defined the same as in the MG.
- Let T be defined using the T in the MG:

$$T(s_t, a_t, s_{t+1}) := T(s_t, \pi_{\theta_1}(s_t), \ldots, a_t, \ldots, \pi_{\theta_m}(s_t), s_{t+1})$$

5.2 Derby Framework

[Source: https://github.com/nkumar15-brown-university/derby]

Derby is a simple bidding, auction, and *market* framework for creating and running auction or market games. Environments in derby can be interfaced in a similar fashion as environments in OpenAI's gym:

```
env = ...
agents = ...
env.init(agents, num_of_days)
for i in range(num_of_trajs):
        all_agents_states = env.reset()
        for j in range(horizon_cutoff):
            actions = []
            for agent in env.agents:
                agent_states = env.get_folded_states(
                                    agent, all_agents_states
                                )
                actions.append(agent.compute_action(agent_states))
            all_agents_states, rewards, done = env.step(actions)
```

A *market* can be thought of as a stateful, repeated auction:

- A market is initialized with m bidders, each of which has a state.
- A market lasts for N days.
- Each day, auction items are put on sale. Each day, the bidders participate in an auction for the available items.
- Each bidder's state is updated at the end of every day. The state can track information such as auction items bought and amount spent.

5.3 Algorithms

The algorithms in this section are derived from a multi-agent perspective. Since this paper only addresses the single-agent scenario, assume $m = 1$ for the algorithms below. The algorithms should fold naturally into their respective canonical single-agent PG algorithms.

5.3.1 Multi-Agent REINFORCE

Definitions

(See Formal RL Definition section)

Assumptions

(See Formal RL Definition section)

Algorithm Derivation

$$
\begin{aligned}
\nabla_{\theta_i} J(\theta_i) &= \nabla_{\theta_i} \int p_\theta(\tau) r^{(i)}(\tau) d\tau \\
&= \int \nabla_{\theta_i} p_\theta(\tau) r^{(i)}(\tau) d\tau \\
&= \int p_\theta(\tau) \nabla_{\theta_i} \log p_\theta(\tau) r^{(i)}(\tau) d\tau \\
&= \int p_\theta(\tau) \nabla_{\theta_i} \log \left[p(s_0) \prod_{t=0}^{T-2} \left(\prod_{j=1}^{m} \pi_{\theta_j}(a_t^{(j)}|s_t) \right) p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)}) \right] r^{(i)}(\tau) d\tau \\
&= \ldots \left[\nabla_{\theta_i} \log p(s_0) + \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\prod_{j=1}^{m} \pi_{\theta_j}(a_t^{(j)}|s_t) \right) + \nabla_{\theta_i} \log p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)}) \right] \ldots \\
&\qquad\qquad \text{(let } \ldots \text{ represent other parts of equation)} \\
&= \ldots \left[0 + \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\prod_{j=1}^{m} \pi_{\theta_j}(a_t^{(j)}|s_t) \right) + 0 \right] \ldots \\
&= \ldots \left[\sum_{t=0}^{T-2} \sum_{j=1}^{m} \nabla_{\theta_i} \log \left(\pi_{\theta_j}(a_t^{(j)}|s_t) \right) \right] \ldots \\
&= \ldots \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)}|s_t) \right) \right] \ldots \\
&= \int p_\theta(\tau) \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)}|s_t) \right) \right] r^{(i)}(\tau) d\tau \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)}|s_t) \right) \right) \left(\sum_{t=0}^{T-2} r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) \right) \right] \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)}|s_t) \right) \left(\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \ldots, a_{t'}^{(m)}) \right) \right] \\
&\qquad\qquad \text{(causality trick)} \\
&= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)}|s_t) \right) \hat{Q}_t^{(i)} \right] \\
&\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)}|s_{j,t}) \right) \hat{Q}_{j,t}^{(i)}
\end{aligned}
$$

using N trajectories sampled from $p_\theta(\tau)$

Algorithm

1. Sample N trajectories from $p_\theta(\tau)$
2. Use trajectories to calculate $\nabla_{\theta_i} J(\theta_i)$
3. $\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J(\theta_i)$

5.3.2 Multi-Agent REINFORCE with baseline

Definitions
(Same as REINFORCE)

Assumptions
(Same as REINFORCE)

Algorithm Derivation

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i} \int p_\theta(\tau) r^{(i)}(\tau) d\tau$$

$$\vdots \quad \text{(see multi-agent REINFORCE derivation for more detail)}$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \hat{Q}_t^{(i)} \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) (\hat{Q}_t^{(i)} - V^{\pi_{\theta_i}}(s_t)) \right]$$

(use baseline to lower variance)

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) (\hat{Q}_{j,t}^{(i)} - V^{\pi_{\theta_i}}(s_{j,t}))$$

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) (\hat{Q}_{j,t}^{(i)} - \hat{V}_{\mathbf{w}_i}(s_{j,t})) \qquad \text{(use VFA)}$$

using N trajectories sampled from $p_\theta(\tau)$

where $\hat{V}_{\mathbf{w}_i}$ is a VFA of parameters $\mathbf{w}_i$ to fit $V^{\pi_{\theta_i}}$

Algorithm

1. Sample N trajectories from $p_\theta(\tau)$
2. Use trajectories to calculate $\nabla_{\theta_i} J(\theta_i)$.

   Use trajectories to calculate $\text{loss}(\hat{V}_{\mathbf{w}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right)^2$,
   where $y_{j,t} = \sum_{t'=t}^{T-2} r^{(i)}(s_{j,t'}, a_{j,t'}^{(1)}, \ldots, a_{j,t'}^{(m)})$.
3. $\theta \leftarrow \theta + \alpha_1 \nabla_{\theta_i} J(\theta_i)$.
   $\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_2 \nabla_{\mathbf{w}_i} \text{loss}(\hat{V}_{\mathbf{w}_i})$.

5.3.3 Multi-Agent Actor-Critic Q with baseline

Definitions
(Same as REINFORCE)

Assumptions
(Same as REINFORCE)

Algorithm Derivation

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i} \int p_\theta(\tau) r^{(i)}(\tau) d\tau$$

$\vdots$ (see multi-agent REINFORCE derivation for more detail)

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \hat{Q}_t^{(i)} \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \left(Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) - V^{\pi_{\theta_i}}(s_t) \right) \right]$$

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left(Q^{\pi_{\theta_i}}(s_{j,t}, a_{j,t}^{(1)}, \dots, a_{j,t}^{(m)}) - V^{\pi_{\theta_i}}(s_{j,t}) \right)$$

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left(\hat{Q}_{\mathbf{u}_i}(s_{j,t}, a_{j,t}^{(1)}, \dots, a_{j,t}^{(m)}) - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right)$$

($\hat{Q}_{\mathbf{u}_i}$ introduces bias, lowers variance)

using N trajectories sampled from $p_\theta(\tau)$

where $\hat{Q}_{\mathbf{u}_i}$ is a VFA of parameters $\mathbf{u}_i$ to fit $Q^{\pi_{\theta_i}}$

where $\hat{V}_{\mathbf{w}_i}$ is a VFA of parameters $\mathbf{w}_i$ to fit $V^{\pi_{\theta_i}}$

Algorithm

1. Sample N trajectories from $p_\theta(\tau)$
2. Use trajectories to calculate $\nabla_{\theta_i} J(\theta_i)$.

   Use trajectories to calculate $\text{loss}(\hat{V}_{\mathbf{w}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right)^2$,
   where $y_{j,t} = \sum_{t'=t}^{T-2} r^{(i)}(s_{j,t'}, a_{j,t'}^{(1)}, \dots, a_{j,t'}^{(m)})$.

   Use trajectories to calculate $\text{loss}(\hat{Q}_{\mathbf{u}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{Q}_{\mathbf{u}_i}(s_{j,t}, a_{j,t}^{(1)}, \dots, a_{j,t}^{(m)}) \right)^2$,
   where $y_{j,t} = \sum_{t'=t}^{T-2} r^{(i)}(s_{j,t'}, a_{j,t'}^{(1)}, \dots, a_{j,t'}^{(m)})$.
3. $\theta \leftarrow \theta + \alpha_1 \nabla_{\theta_i} J(\theta_i)$.
   $\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_2 \nabla_{\mathbf{w}_i} \text{loss}(\hat{V}_{\mathbf{w}_i})$.
   $\mathbf{u}_i \leftarrow \mathbf{u}_i + \alpha_3 \nabla_{\mathbf{u}_i} \text{loss}(\hat{Q}_{\mathbf{u}_i})$.

5.3.4 Multi-Agent Actor-Critic TD (has baseline)

Definitions
(Same as REINFORCE)

Assumptions
(Same as REINFORCE)

Multi-Agent Bellman Rollout Derivation

$$
\begin{aligned}
V^{\pi_{\theta_i}}(s_t) :&= \mathbb{E}_{\tau\sim p_\theta(\tau_t)}\left[\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)}) \middle| s_t\right] \\
&= \mathbb{E}_{a_{t+1}^{(1)},\dots,a_{t+1}^{(m)}\sim\pi_\theta(a_{t+1}^{(1)},\dots,a_{t+1}^{(m)}|s_{t+1})}\left[\mathbb{E}_{\tau\sim p_\theta(\tau_t)}\left[\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)}) \middle| s_t, a_t^{(1)}, \dots, a_t^{(m)}\right] \middle| s_t\right] \\
&= \mathbb{E}_{a_{t+1}^{(1)},\dots,a_{t+1}^{(m)}\sim\pi_\theta(a_{t+1}^{(1)},\dots,a_{t+1}^{(m)}|s_{t+1})}\left[Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) \middle| s_t\right]
\end{aligned}
$$

And:

$$
\begin{aligned}
Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) :&= \mathbb{E}_{\tau\sim p_\theta(\tau_t)}\left[\sum_{t'=t}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)}) \middle| s_t, a_t^{(1)}, \dots, a_t^{(m)}\right] \\
&= \mathbb{E}_{\tau\sim p_\theta(\tau_t)}\left[r^{(i)}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) + \sum_{t'=t+1}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)}) \middle| s_t, a_t^{(1)}, \dots, a_t^{(m)}\right] \\
&= r^{(i)}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) + \mathbb{E}_{\tau\sim p_\theta(\tau_t)}\left[\sum_{t'=t+1}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)}) \middle| s_t, a_t^{(1)}, \dots, a_t^{(m)}\right] \\
&= r^{(i)}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) + \mathbb{E}_{\tau\sim p_\theta(\tau_{t+1})}\left[\sum_{t'=t+1}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)})\right] \\
&= \cdots \mathbb{E}_{\tau\sim p_\theta(\tau_{t+1})}\left[\sum_{t'=t+1}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \dots, a_{t'}^{(m)})\right] \cdots \\
&\qquad\qquad \text{(Let } \dots \text{ represent other parts of equation)}
\end{aligned}
$$

$$= \cdots \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)})} \Bigg[\mathbb{E}_{a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)} \sim \pi_\theta(a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)}|s_{t+1})} \left[\mathbb{E}_{\tau \sim p_\theta(\tau_{t+1})} \left[\sum_{t'=t+1}^{T-2} r^{(i)}(s_{t'}, a_{t'}^{(1)}, \ldots, a_{t'}^{(m)}) \middle| s_{t+1}, a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)} \right] \middle| s_{t+1} \right] \Bigg] \cdots$$

$$= \cdots \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)})} \left[\mathbb{E}_{a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)} \sim \pi_\theta(a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)}|s_{t+1})} \left[Q^{\pi_{\theta_i}}(s_{t+1}, a_{t+1}^{(1)}, \ldots, a_{t+1}^{(m)}) \middle| s_{t+1} \right] \right] \cdots$$

$$= \cdots \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)})} \left[V^{\pi_{\theta_i}}(s_{t+1}) \right] \cdots$$

$$= r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t^{(1)}, \ldots, a_t^{(m)})} \left[V^{\pi_{\theta_i}}(s_{t+1}) \right]$$

AdX Research Summary

Algorithm Derivation

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i} \int p_\theta(\tau) r^{(i)}(\tau) d\tau$$

$\vdots$ (see multi-agent REINFORCE derivation for more detail)

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \hat{Q}_t^{(i)} \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \left(Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) - V^{\pi_{\theta_i}}(s_t) \right) \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \left(r^{(i)}(s_t, a_t^{(1)}, \ldots, a_t^{(m)}) + \mathbb{E}\left[V^{\pi_{\theta_i}}(s_{t+1}) \right] - V^{\pi_{\theta_i}}(s_t) \right) \right]$$

(bootstrapping)

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left[r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \ldots, a_{j,t}^{(m)}) + V^{\pi_{\theta_i}}(s_{j,t+1}) - V^{\pi_{\theta_i}}(s_{j,t}) \right]$$

(single sample bootstrapping)

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left[r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \ldots, a_{j,t}^{(m)}) + \hat{V}_{\mathbf{w}_i}(s_{j,t+1}) - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right]$$

(use VFA; VFA in one-step rollout introduces bias, lowers variance)

using N trajectories sampled from $p_\theta(\tau)$

where $\hat{V}_{\mathbf{w}_i}$ is a VFA of parameters $\mathbf{w}_i$ to fit $V^{\pi_{\theta_i}}$

Algorithm

1. Sample N trajectories from $p_\theta(\tau)$
2. Use trajectories to calculate $\nabla_{\theta_i} J(\theta_i)$.

   Use trajectories to calculate $\text{loss}(\hat{V}_{\mathbf{w}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right)^2$,
   where $y_{j,t} = r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \ldots, a_{j,t}^{(m)}) + \hat{V}_{\mathbf{w}_i}(s_{j,t+1})$.
3. $\theta \leftarrow \theta + \alpha_1 \nabla_{\theta_i} J(\theta_i)$.
   $\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_2 \nabla_{\mathbf{w}_i} \text{loss}(\hat{V}_{\mathbf{w}_i})$.

5.3.5 Multi-Agent Actor-Critic SARSA with baseline

Definitions
(Same as REINFORCE)

Assumptions
(Same as REINFORCE)

Multi-Agent Bellman Rollout Derivation
(SAME as AC TD)

Algorithm Derivation

$$\nabla_{\theta_i} J(\theta_i) = \nabla_{\theta_i} \int p_\theta(\tau) r^{(i)}(\tau) d\tau$$

$\vdots$ (see multi-agent REINFORCE derivation for more detail)

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \hat{Q}_t^{(i)} \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_t^{(i)} | s_t) \right) \left(Q^{\pi_{\theta_i}}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) - V^{\pi_{\theta_i}}(s_t) \right) \right]$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\dots \left(r^{(i)}(s_t, a_t^{(1)}, \dots, a_t^{(m)}) + \mathbb{E} \left[\mathbb{E} \left[Q^{\pi_{\theta_i}}(s_{t+1}, a_{t+1}^{(1)}, \dots, a_{t+1}^{(m)}) \right] \right] - V^{\pi_{\theta_i}}(s_t) \right) \right]$$

(bootstrapping; using $\dots$ to represent preceding parts of equation)

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left[r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \dots, a_{j,t}^{(m)}) + Q^{\pi_{\theta_i}}(s_{j,t+1}, a_{j,t+1}^{(1)}, \dots, a_{j,t+1}^{(m)}) - V^{\pi_{\theta_i}}(s_{j,t}) \right]$$

(single sample bootstrapping)

$$\approx \frac{1}{N} \sum_{j=0}^{N-1} \sum_{t=0}^{T-2} \nabla_{\theta_i} \log \left(\pi_{\theta_i}(a_{j,t}^{(i)} | s_{j,t}) \right) \left[r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \dots, a_{j,t}^{(m)}) + \hat{Q}_{\mathbf{u}_i}(s_{j,t+1}, a_{j,t+1}^{(1)}, \dots, a_{j,t+1}^{(m)}) - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right]$$

(use VFA; $\hat{Q}_{\mathbf{u}_i}$ introduces bias, lowers variance)

using N trajectories sampled from $p_\theta(\tau)$

where $\hat{Q}_{\mathbf{u}_i}$ is a VFA of parameters $\mathbf{u}_i$ to fit $Q^{\pi_{\theta_i}}$

where $\hat{V}_{\mathbf{w}_i}$ is a VFA of parameters $\mathbf{w}_i$ to fit $V^{\pi_{\theta_i}}$

Algorithm

1. Sample N trajectories from $p_\theta(\tau)$
2. Use trajectories to calculate $\nabla_{\theta_i} J(\theta_i)$.

Use trajectories to calculate $\text{loss}(\hat{V}_{\mathbf{w}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{V}_{\mathbf{w}_i}(s_{j,t}) \right)^2$,
where $y_{j,t} = \sum_{t'=t}^{T-2} r^{(i)}(s_{j,t'}, a_{j,t'}^{(1)}, \ldots, a_{j,t'}^{(m)})$.

Use trajectories to calculate $\text{loss}(\hat{Q}_{\mathbf{u}_i}) = \sum_{j,t} \left(y_{j,t} - \hat{Q}_{\mathbf{u}_i}(s_{j,t}, a_{j,t}^{(1)}, \ldots, a_{j,t}^{(m)}) \right)^2$,
where $y_{j,t} = r^{(i)}(s_{j,t}, a_{j,t}^{(1)}, \ldots, a_{j,t}^{(m)}) + \hat{Q}_{\mathbf{u}_i}(s_{j,t+1}, a_{j,t+1}^{(1)}, \ldots, a_{j,t+1}^{(m)})$.

3. $\theta \leftarrow \theta + \alpha_1 \nabla_{\theta_i} J(\theta_i)$.
   $\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_2 \nabla_{\mathbf{w}_i} \text{loss}(\hat{V}_{\mathbf{w}_i})$.
   $\mathbf{u}_i \leftarrow \mathbf{u}_i + \alpha_3 \nabla_{\mathbf{u}_i} \text{loss}(\hat{Q}_{\mathbf{u}_i})$.