MASAC vs ISAC

Project assumptions

The project aims to evaluate and compare the performance of ISAC and MASAC algorithms in various multi-agent reinforcement learning (MARL) environments, focusing on scenarios that emphasize cooperation, competition, and mixed dynamics. The core assumption is that ISAC's independent learning structure and entropy-regularized exploration will provide advantages in adaptability and stability compared to MASAC's shared-environment optimization. Additionally, Melting Pot's diverse scenarios will help test generalization capabilities effectively. ## Project environmment ### Software #### BenchMARL: Multi-Agent Reinforcement Learning Library BenchMARL is Multi-Agent Reinforcement Learning (MARL) library focused on standardized benchmarking across a variety of algorithms and environments. Its mission is to present a standardized interface that allows easy integration of new algorithms and environments to provide a fair comparison with existing solutions. Core design principles include:

- **Reproducibility**: Achieved through systematic grounding and standardization of configuration.
- **Standardized Reporting**: Strong plotting and reporting.
- **Modular Experimentation**: Experiments that are independent of the algorithm, environment, and model choices.
- **Broad MARL Ecosystem Coverage**: Breadth over the MARL ecosystem.
- **Ease of Integration**: Easy implementation of new algorithms, environments, and models.
- **TorchRL Integration**: Leveraging the know-how and infrastructure of TorchRL, without reinventing the wheel.

MeltingPot Melting Pot evaluates how well agents generalize to new social environments, including interactions with both known and unfamiliar individuals. It was designed to test a broad range of social interactions including cooperation, competition, reciprocation, stubbornness and trust. Melting Pot offers researchers a set of over 50 multi-agent reinforcement learning substrates (multi-agent games) and over 256 unique test scenarios for evaluation. The performance of agents on test scenarios allows to assess agents in:

- **Adaptability**: Perform well across a range of social situations where individuals are interdependent.
- **Interpersonal Generalization**: Interact effectively with unfamiliar individuals not seen during training.

Technical realisation

Python scripts for experiments and Notebooks for visualisation

Algorithms to be implemented and tested

ISAC

Soft Actor-Critic (SAC) is a model-free RL algorithm that combines off-policy learning with entropy regularization to encourage exploration and stabilize learning. It extends SAC for multi-agent scenarios, allowing agents to independently optimize their policies while optionally sharing critic parameters within agent groups. The algorithm uses entropy to measure policy randomness and incorporates it into the loss function. The policy induction process in the ISAC algorithm involves constructing a probabilistic policy tailored to the action space—continuous or discrete—by leveraging neural networks to map states to probability distributions over actions. For continuous actions, the policy generates parameters (mean and scale) for distributions like IndependentNormal or TanhNormal, ensuring bounded actions when needed. For discrete actions, logits are passed into distributions such as Categorical or MaskedCategorical, accommodating optional action constraints. The policy is trained to maximize the soft Q-function while encouraging exploration through entropy regularization, with the trade-off governed by an entropy coefficient (alpha) that can be fixed or learned.

MASAC: Multi Agent adaptation of Soft Actor Critic Reinforcement Learning Algorithm

MASAC is an extension of the Soft Actor-Critic reinforcement learning algorithm, adapted for multi-agent environments. MASAC builds on the SAC approach by allowing multiple agents to learn simultaneously in a shared environment. This algorithm is based on Centralized Training with Decentralized Execution (CTDE) scheme often used in MARL, where agents are trained using centralized critic with global information, such as the states and actions of all agents. However, during action execution, each agent acts independently using only its local observations. Additionally, MASAC can

be modified to utilize Lyapunov stability in orfer to ensure that the learned policies and system dynamics remain stable and safe during multi-agent interactions.

Necessary libraries

- PyTorch: For deep learning model implementations.
- TorchRL: For reinforcement learning utilities and environment handling.
- BenchMARL: To provide the MARL framework for standardized benchmarking.
- Melting Pot: For diverse testing substrates and social interaction scenarios.
- NumPy and Matplotlib: For data processing and visualization.
- Seaborn: For advanced statistical data visualizations in the analysis phase.
- Jupyter Notebooks: For interactive experimentation and result presentation.

Experiment propositions

Collaborative Cooking - Cooperative

In this scenario, agents work together in a kitchen setting to prepare and serve food items. They must coordinate their actions to complete tasks efficiently, such as gathering ingredients, cooking, and serving dishes within a time limit. Success depends on effective teamwork and sharing resources. This environment encourages agents to optimize collective outcomes, making it an example of a cooperative problem

Chicken in the matrix: Arena - Competitive

Individuals can gather resources of different colors. Players' encounters are resolved with thesame payout matrix as the game 'Chicken', in which both players attempting to take advantage of the other leads to the worst outcome for both. Collecting red resources pushes one's strategy choice toward playing 'hawk' while collecting green resources pushes it toward playing 'dove'.

Clean Up - Mixed (Primarily Cooperative with Competitive Elements)

Clean Up is a seven player game. Players are rewarded for collecting apples (reward +1). In Clean Up, apples grow in an orchard at a rate inversely

related to the cleanliness of a nearby river. The river accumulates pollution at a constant rate. Beyond a certain threshold of pollution, the apple growth rate in the orchard drops to zero. Players have an additional action allowing them to clean a small amount of pollution from the river. However, the cleaning action only works on pollution within a small distance in front of the agents, requiring them to physically leave the apple or chard to clean the river. Thus, players maintain a public good of orchard regrowth through effortful contributions. Players are also able to zap others with a beam that removes any player hit from the game for 50 steps.