Technical Explanation of Federated Feudal Reinforcement Learning Implementation

1 Hierarchical (Feudal) Reinforcement Learning in Privacy-Oriented Context

1.1 Overview

- The implementation adopts a feudal (or hierarchical) reinforcement learning paradigm.
- A single global **ManagerAgent** sets high-level goals based on the overall (global) state.
- Each client operates multiple **WorkerAgents** in a local environments (in this case, a variant of the Catch game) that share a single network, ensuring that local updates reflect a common policy.
- These worker networks are later synchronized using **federated averaging** [1][1] after local training rounds.
- The manager-worker structure is similar to the architecture described in *FeUdal Networks for Hierarchical Reinforcement Learning* by Vezhnevets et al. (2017) and builds on earlier work by Dayan and Hinton (1993).

1.2 Manager Agent

- Computes sub-goals (e.g., target positions) from a global state that aggregates information from all workers and the target.

1.3 Worker Agent

- Uses a combined state (its own observation plus the goal provided by the manager) to sample actions that move it toward achieving the manager's intent.

1.4 Mathematical Formulation for the Manager Network

Given a global state vector s, the network computes:

(1) **First Hidden Layer:**

$$h_1 = \text{ReLU}(W_1 \cdot s + b_1)$$

The input state s is multiplied by a weight matrix W_1, added to a bias vector b_1, and passed through the ReLU activation function (which sets negative values to zero) to produce the first hidden layer output h_1.

(2) **Second Hidden Layer:**

$$h_2 = \text{ReLU}(W_2 \cdot h_1 + b_2)$$

The output h_1 is transformed using weights W_2 and bias b_2, then passed through ReLU to yield h_2.

[1] `https://arxiv.org/abs/1602.05629`

(3) **Manager Mean Output (Goal Generation):**

$$\mu_{\text{manager}} = \tanh(W_\mu \cdot h_2 + b_\mu) \times \text{scale}$$

A linear transformation of h_2 using weights W_μ and bias b_μ is passed through a tanh activation to produce values between -1 and 1. Multiplication by a scaling factor adjusts the output to the environment's coordinate range.

(4) **Manager Standard Deviation Output:**

$$\sigma_{\text{manager}} = \text{softplus}(W_\sigma \cdot h_2 + b_\sigma) + \epsilon$$

A similar transformation is applied to h_2. The softplus function ensures that the standard deviation is positive, and a small constant ϵ (e.g., 10^{-5}) is added for numerical stability.

1.5 Mathematical Formulation for the Worker Network

For an augmented state s_{worker} (concatenating the worker's observation with the goal), the computations are similar:

(1) **First Hidden Layer:**

$$h'_1 = \text{ReLU}(W'_1 \cdot s_{\text{worker}} + b'_1)$$

(2) **Second Hidden Layer:**

$$h'_2 = \text{ReLU}(W'_2 \cdot h'_1 + b'_2)$$

(3) **Worker Mean Output:**

$$\mu_{\text{worker}} = W'_\mu \cdot h'_2 + b'_\mu$$

(4) **Worker Standard Deviation Output:**

$$\sigma_{\text{worker}} = \text{softplus}(W'_\sigma \cdot h'_2 + b'_\sigma) + \epsilon$$

For further details on hierarchical goal setting, see Vezhnevets et al. (2017).

2 Actor-Critic and Advantage Estimation

2.1 Overview

The training employs an actor-critic method for both manager and workers. Two networks are maintained per agent:

- **Policy (Actor):** Outputs a parameterized Gaussian distribution over actions or goals.
- **Value (Critic):** Estimates the state value function $V(s)$.

2.2 Generalized Advantage Estimation (GAE)

- **Temporal-Difference (TD) Error:**

$$\delta_t = r_t + \gamma \cdot V(s_{t+1}) - V(s_t)$$

At time step t, the TD error δ_t measures the difference between the observed reward r_t plus the discounted value of the next state $V(s_{t+1})$ and the current state's value $V(s_t)$.

Given a value function defined as:

$$V(s_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \, r_{t+k} \,\middle|\, s_t\right]$$

where:

- r_{t+k} is the reward received k steps in the future,
- γ is the discount factor (a number between 0 and 1) that weighs future rewards,
- The expectation is taken over the stochastic transitions of the environment and the actions dictated by the current policy.

- **Advantage Calculation:**

$$A_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \cdot \delta_{t+l}$$

The advantage A_t at time t is computed by summing future TD errors, each discounted exponentially by the factor $(\gamma\lambda)$, where γ is the discount factor and λ is a smoothing parameter.

2.3 Loss Functions

- **Actor Loss:**

$$\mathcal{L}_{\text{actor}} = -\log \pi(a_t|s_t) \cdot A_t - \beta \cdot H(\pi(\cdot|s_t))$$

This loss includes:

- $-\log \pi(a_t|s_t) \cdot A_t$: Encourages actions that yield higher advantages.
- $-\beta \cdot H(\pi(\cdot|s_t))$: An entropy bonus (weighted by β) that promotes exploration by preventing the policy from becoming too confident.

- **Critic Loss:**

$$\mathcal{L}_{\text{critic}} = \frac{1}{2} \left(V(s_t) - R_t\right)^2$$

This is the mean squared error between the estimated value $V(s_t)$ and the target return R_t computed via GAE.

For further details on GAE and actor-critic updates, see Schulman et al. (2016).

3 Federated Learning Components

3.1 Federated Averaging

After each communication round, the worker networks from all clients are averaged to form a new global model. The formula is:

$$\theta_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \theta_i$$

where θ_i represents the parameters of the i-th client's worker network, and N is the number of clients. (See McMahan et al., 2016.)

3.2 FedProx Regularization

To address heterogeneity in client data and stabilize updates, a **FedProx** term is added:

$$\mathcal{L}_{\text{prox}} = \frac{\mu}{2} \left\|\theta_i - \theta_{\text{global}}\right\|^2$$

Here, μ is a coefficient that controls the strength of the penalty for deviations between a client's parameters θ_i and the global model θ_{global}. In our implementation, the coefficient μ is scheduled dynamically, increasing linearly from 0 to a maximum value over a defined number of episodes [2][2]. This gradual enforcement helps maintain consistency between the client models and the global model.

[2] `https://arxiv.org/abs/1812.06127`

4 Curriculum Learning and Intrinsic Rewards

4.1 Curriculum Weighting

The approach gradually shifts the workers' reliance from the true target to the manager's goal:

$$g_{\text{worker}} = (1 - \alpha) \cdot g_{\text{true}} + \alpha \cdot g_{\text{manager}}$$

The curriculum weight α starts near 0 and increases to 1 over training episodes, blending the true target g_{true} with the manager's goal g_{manager}. (See Bengio et al., 2009.)

4.2 Intrinsic Rewards

In addition to extrinsic rewards, workers receive an intrinsic reward based on their proximity to the assigned goal:

$$r_{\text{intrinsic}} = -\kappa \cdot \|p - g\|$$

where p is the worker's current position, g is the goal, and κ is a scaling coefficient. The reward becomes less negative as the worker approaches the goal.

5 Additional Techniques

5.1 Gradient Clipping

Gradient clipping is applied to both manager and worker networks to prevent exploding gradients. A maximum norm (e.g., 5.0) is enforced; if the gradient exceeds this norm, it is scaled down. (See Pascanu et al., 2013.)

5.2 Environment and Randomization

- The environment is a variant of the Catch game.
- Randomization functions are used to:
  - **Drone Initialization:** Drone positions are sampled with a minimum separation constraint to avoid overlapping positions.
  - **Target Placement:** The target is placed at a location ensuring a specified minimum separation from all drones.
  - **Dynamic Target Speed:** The target speed is randomized, adding variability and challenge to the task.
- This diversity in initialization promotes robustness in learning.

6 Consistency Loss Between Manager and Workers

An optional consistency loss is incorporated once the curriculum weight exceeds a specified threshold. This loss minimizes the discrepancy between the manager's predicted sub-goals and the desired goals—computed as the difference between the true target and the worker's current position. Such alignment reinforces coherent high-level guidance with local worker behavior.

7 Learning Rate Scheduling

Both the manager and worker networks utilize learning rate schedulers to decay the learning rate periodically (e.g., by a factor of 0.9 every fixed number of episodes). This scheduling strategy contributes to a more stable training process by gradually reducing the learning rate as training progresses.

8 Summary of the Pipeline

- **Network Architectures:**
  - **Manager Network:** Processes the global state and generates Gaussian parameters (mean and standard deviation) for sub-goals.
  - **Worker Network:** Processes a concatenated vector (own state + goal) and produces Gaussian parameters for action sampling.
- **Training Dynamics:**
  - **Actor-Critic Losses:** Both manager and worker losses are computed using GAE, entropy regularization, and mean squared error (MSE) for the critic.
  - **FedProx Regularization:** A proximal term is added to the worker loss to maintain consistency with the global model.
  - **Federated Averaging:** Worker network parameters are averaged across clients after local training rounds.
- **Hierarchical and Curriculum Learning:**
  - Manager goals are blended with the true target based on a curriculum schedule.
  - Intrinsic rewards encourage workers to minimize the distance to their sub-goals.

9 Citations

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2016). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. `https://arxiv.org/abs/1602.05629`.

[2] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). *Federated Optimization in Heterogeneous Networks*.

[3] Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). *FeUdal Networks for Hierarchical Reinforcement Learning*.

[4] Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*.

[5] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). *Curriculum Learning*.

[6] Pascanu, R., Mikolov, T., & Bengio, Y. (2013). *On the Difficulty of Training Recurrent Neural Networks*.