Adaptive Regret Minimization for Multi-Task Agents via Reactive Hedge

Team: PAC-srsk-1729

Sarvagya Porwal
sarvp2486@gmail.com

Abstract

We introduce a "Tribe of Experts" architecture for decision-making in multi-task reinforcement learning environments. To effectively ensemble a diverse population of agents (Entities) trained with varying hyperparameters, we propose a **Reactive Hedge** algorithm. This method minimizes cumulative regret during inference by dynamically weighting experts based on a normalized Temporal Difference (TD) error proxy. Furthermore, we implement an adaptive learning rate mechanism and a refined nucleus sampling strategy to balance exploration and stability. This report details the theoretical framework, the practical implementation, and a retrospective analysis of the competition strategy. The code for this implementation is available at `https://github.com/Sar2580P/Pokemon-Challenge-track1-NeurIPS2025-competition`.

1 Methodology

Our approach shifts from relying on a single monolithic policy to maintaining a population of experts, $\mathcal{E} = \{E_1, \ldots, E_N\}$. To maximize performance without ground-truth optimal labels during inference, we employ a regret minimization framework inspired by the Hedge algorithm [2], adapted for the volatility of RL dynamics.

1.1 The Reactive Hedge Algorithm

The core mechanism maintains a probability distribution (weights) w_t over the expert population. At each time step t, the weight of expert i is updated based on a calculated loss $L_{i,t}$:

$$w_{i,t+1} = \frac{w_{i,t} \cdot \exp(-\eta_{i,t} L_{i,t})}{\sum_{j=1}^{N} w_{j,t} \cdot \exp(-\eta_{j,t} L_{j,t})} \tag{1}$$

where $\eta_{i,t}$ is an adaptive learning rate. The crucial innovation lies in the design of the loss function L, which must serve as a proxy for regret in the absence of supervision.

1.2 Loss Design: Normalized TD-Error Proxy

We utilize the Temporal Difference (TD) error as a real-time performance metric. To stabilize the exponential updates of the Hedge algorithm, we employ a squashed, normalized loss:

$$\delta_{i,t} = (r_t + \gamma_i V_i(s_{t+1})) - V_i(s_t) \tag{2}$$

$$z_{i,t} = \frac{\delta_{i,t}}{\sigma_{\text{popart}} + \epsilon} \tag{3}$$

$$L_{i,t} = -\tanh(z_{i,t}) \tag{4}$$

Here, σ_{popart} is the standard deviation derived from a PopArt normalization layer.

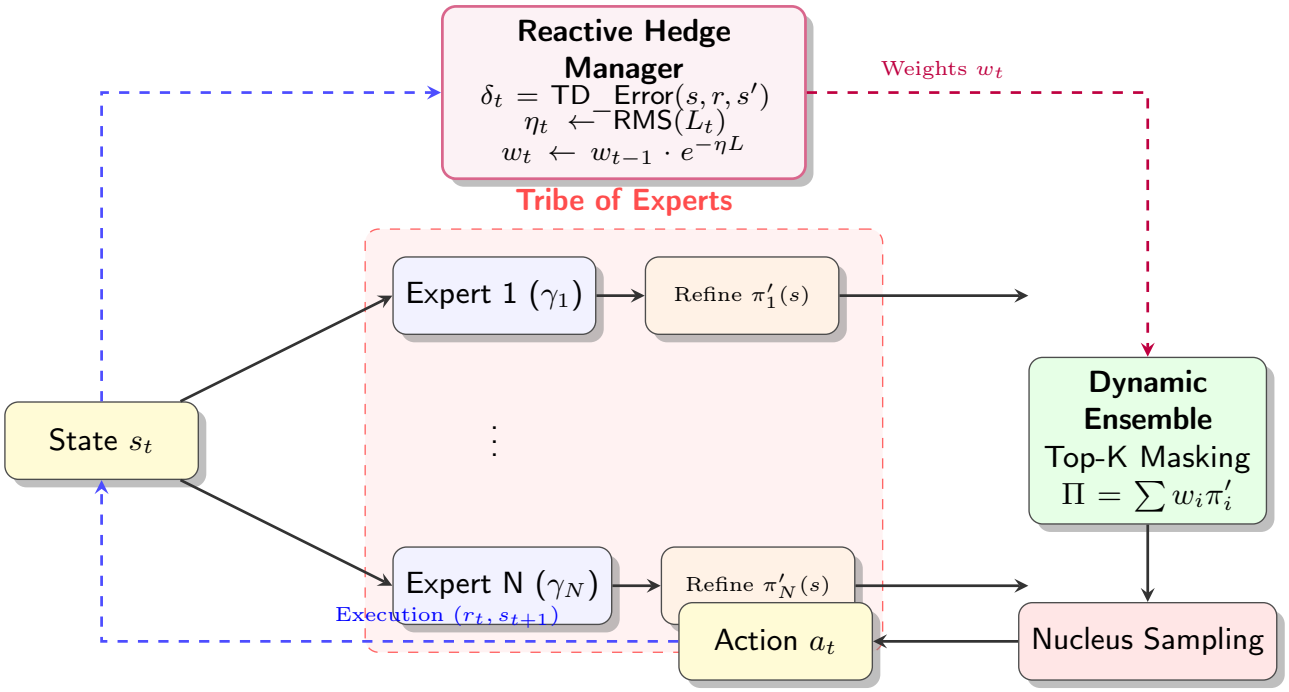

Figure 1: Architecture Diagram of the Adaptive Tribe Agent. The system processes state s_t through multiple experts in parallel. The **Reactive Hedge Manager** dynamically updates expert weights w_t based on TD-error feedback, modulating the contribution of each expert to the final **Dynamic Ensemble**.

1.3 Adaptive Learning Rate (η-Adaptability)

A fixed learning rate for the Hedge update is prone to instability in RL. While previous work has proposed adapting η based on the mixability gap [1], we implement an RMS-Prop style adaptation to dampen volatility. We track the running average of squared losses $G_{i,t}$ and scale η dynamically:

$$\eta_{i,t} = \frac{\eta_{\text{base}}}{\sqrt{G_{i,t}} + \epsilon} \tag{5}$$

2 Proposed Evaluation Metrics

To analyze the internal dynamics of the tribe and the confidence of the resulting policy, we employ the following metrics. **Note on Computation:** As the primary objective during competition is latency minimization, these metrics are computed via **post-hoc analysis**. We replay the recorded observation sequences from the competition episodes through the frozen agent architecture ("Shadow Mode"). At each timestep t, we feed the recorded state s_t to the experts to extract internal values (π, V, w) without altering the trajectory.

2.1 Policy Uncertainty (Action Entropy)

We measure the uncertainty of the final ensembled policy Π_{final} using Shannon entropy. High entropy indicates the agent is uncertain or exploring, while low entropy indicates high confidence in a specific action.

$$H_\pi(s_t) = -\sum_{a \in \mathcal{A}} \Pi_{\text{final}}(a|s_t) \log \Pi_{\text{final}}(a|s_t) \tag{6}$$

2.2 Expert Diversity (Weight Entropy)

To determine if the tribe is benefiting from the "wisdom of crowds" or collapsing into a "dictatorship" (where one expert dominates), we calculate the entropy of the expert weights w_t.

$$H_w(t) = -\sum_{i=1}^{N} w_{i,t} \log w_{i,t} \tag{7}$$

- **High H_w:** Indicates a democratic ensemble where many experts contribute to the decision.
- **Low H_w:** Indicates mode collapse, where the system relies heavily on a single expert.

2.3 Confidence Margin

We quantify the decisiveness of the agent by the probability mass assigned to the top selected action relative to the second best.

$$C_{\text{margin}}(s_t) = \Pi(a_{\text{best}}|s_t) - \Pi(a_{\text{second}}|s_t) \tag{8}$$

3 Implementation

The strategy is implemented within the `InferenceTribeMTA` class, which manages the lifecycle of multiple `InferenceEntityMTA` instances.

3.1 Inference Loop and Dynamic Ensembling

The inference process does not simply average expert outputs. It employs a multi-stage refinement process to ensure state-space coverage and prevent mode collapse:

1. **Per-Gamma Refinement:** Each expert i refines its policy $\pi_i(a|s)$ using its internal Advantage estimate $A_i(s, a)$. This acts as a soft pruning step:

$$\pi'_i(a|s) \propto \pi_i(a|s) \cdot \text{softmax}\left(\frac{A_i(s, a)}{T}\right) \tag{9}$$

2. **Top-K Masking:** To reduce noise from poor-performing experts, we apply a Top-K filter to the global weights w_t. Experts with weights below the K-th largest value are masked to zero, and the remaining weights are re-normalized.

3. **Nucleus Sampling:** The final ensembled distribution $\Pi_{\text{final}} = \sum w'_i \pi'_i$ is sampled using Nucleus (Top-p) sampling. This truncates the long tail of the distribution, accumulating the smallest set of actions whose probability mass exceeds threshold $p = 0.9$.

3.2 Algorithm

The complete logic flow, including the interaction between the inference step and the reactive weight update, is detailed in Algorithm 1.

Algorithm 1 Adaptive Tribe Agent with Reactive Hedge

Input: Experts $\mathcal{E} = \{E_1, \ldots, E_N\}$, Top-$K$ param K, Decay β, Base rate η_{base}.
Initialize: Weights $w_i \leftarrow 1/N$, Gradient trackers $G_i \leftarrow 0 \; \forall i$.
while Episode not done **do**
    $s_t \leftarrow \text{GetObservation}()$
    // Phase 1: Multi-Expert Inference
    for each expert $E_i \in \mathcal{E}$ **do**
        $\pi_i, V_i \leftarrow E_i.\text{Forward}(s_t)$
        $A_i \leftarrow \text{CalcAdvantage}(s_t)$
        $\pi'_i \leftarrow \text{Softmax}(A_i/T) \odot \pi_i$ ▷ Refinement
    end for
    // Phase 2: Dynamic Ensembling
    $w' \leftarrow \text{MaskTopK}(w, K)$ ▷ Filter noise
    $\Pi_{final} \leftarrow \sum_{i=1}^{N} w'_i \cdot \pi'_i$
    Calculate Metrics:
    $H_\pi \leftarrow -\sum \Pi_{final} \log \Pi_{final}$ ▷ Policy Entropy
    $H_w \leftarrow -\sum w \log w$ ▷ Expert Diversity
    $a_t \sim \text{NucleusSample}(\Pi_{final}, p = 0.9)$
    Execute a_t, observe r_t, s_{t+1}, d_t
    // Phase 3: Reactive Hedge Adaptation
    for each expert $E_i \in \mathcal{E}$ **do**
        if d_t is False **then**
            $\delta_i \leftarrow (r_t + \gamma_i V_i(s_{t+1})) - V_i(s_t)$ ▷ TD Error
            $z_i \leftarrow \delta_i / (\sigma_{\text{popart}} + \epsilon)$ ▷ Normalize
            $\mathcal{L}_i \leftarrow -\tanh(z_i)$ ▷ Proxy Loss
            $G_i \leftarrow \beta G_i + (1 - \beta)\mathcal{L}_i^2$ ▷ Update RMS
            $\eta_i \leftarrow \eta_{base} / (\sqrt{G_i} + \epsilon)$ ▷ Adapt η
            $w_i \leftarrow w_i \cdot \exp(-\eta_i \cdot \mathcal{L}_i)$ ▷ Update Weight
        end if
    end for
    $w \leftarrow w / \sum w$ ▷ Normalize Weights
end while

4 Competition Results

The proposed architecture was deployed under the username **PAC-srsk-1729**. Table 1 summarizes the final rankings achieved in the competition tracks.

Table 1: Final Competition Standings (Track 1)

Track Category	Rank	Notes
Track 1 - Gen1ou	**3**	High stability observed in expert ensemble.
Track 1 - Gen9ou	**7**	Impacted by team composition meta-game.

Additional Optimization: To achieve these results, the base experts were further fine-tuned using parsed replays specifically filtered for the targeted Elo range. This ensured the imitation learning targets were relevant to high-level competitive play.

5 Retrospective Analysis: Challenges & Mistakes

Despite the strong performance, the development process was hindered by two significant strategic missteps.

5.1 Mistake 1: Underestimating Team Design

I initially operated under the assumption that a robust policy could fully compensate for suboptimal team composition. I treated "team design" lightly, believing that reinforcement learning could generalize across any loadout. The flaw in this reasoning became apparent in the Gen9ou track, where our agent was consistently defeated by `PAC-ED-Testing`. Their strategy utilized a single, highly optimized team choice across all matches, demonstrating that in Pokémon, team synergy provides a foundational advantage that pure policy cannot easily overcome.

5.2 Mistake 2: The Recurrent State Space Model (HRM) Rabbit Hole

Approximately 75% of the total competition timeline was invested in developing an alternative Recurrent State Space Model (RSSM), inspired by the Hierarchical Reasoning Model (HRM) [3], to replace the standard Trajectory Encoder. Early metrics were promising, leading to a "sunk cost" fallacy where more compute and time were poured into an architecture that ultimately failed to converge within the constraints.

The core failure mode was the computational complexity required for the iterative reasoning steps. The architecture relied on a nested loop of Mixer, Low-Level (L), and High-Level (H) model updates:

$$\begin{aligned}
&\textbf{For } h \in 1 \dots H: \\
&\quad \overline{z}_H = f_{\text{Mixer}}(z_H, \tilde{x}_t) \\
&\quad \textbf{For } l \in 1 \dots L: \\
&\qquad z_L = f_{\text{L-Model}}(z_L, \overline{z}_H) \\
&\quad z_H = f_{\text{H-Model}}(z_L, z_H) \\
&\gamma_t = \text{mean}(z_H) \quad \text{(Reasoning Dim)}
\end{aligned} \tag{10}$$

Even with chunked attention, the complexity scaled as $O(L \cdot H \cdot N_{\text{chunks}})$. In my implementation, I restricted $L, H \approx 4, 5$ to maintain throughput. However, a post-mortem review of the original HRM literature revealed that successful implementations typically require large iteration counts ($L \cdot H > 100$) to ensuring the convergence of the latent states (z_H, z_L), rather than merely capturing long-range dependencies. This computational bottleneck was realized too late. Consequently, I had to abandon this extensive "rabbit hole" exploration and pivot rapidly to refining the inference-time model to meet the competition deadline.

6 Future Work

While the current ideation phase for optimized team design is complete, we were unable to implement it due to time constraints. Future iterations will prioritize integrating team-building logic directly with the policy. Furthermore, recognizing the potential of the HRM architecture which has achieved high performance in Arc-AGI benchmarks, we plan to conduct a deep dive into the convergence properties of HRM latent states to make it viable for long-sequence game playing.

7 Acknowledgments

I would like to extend my gratitude to the competition organizers for hosting this challenging event. A special thanks to **Jake Grigsby** for his detailed guidance and support throughout the competition.

References

[1] Van Erven, T., Koolen, W. M., Grünwald, P., & De Rooij, S. (2011). Adaptive hedge. *Advances in Neural Information Processing Systems*, 24.

[2] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119–139.

[3] Ren, Y., et al. (2024). HRM: Hierarchical Reasoning Model. *International Conference on Learning Representations*.