Dynamic Sunrise

James Thompson

May 29, 2025

1 Introduction

The idea of Dynamic sunrise is to use a dynamic ensemble of SAC actors. The dynamic nature of the ensemble will help in two aspects by speeding up exploration early on in the training as well as allowing the ensemble to "prune" ineffective base learners. Furthermore it can help the ensemble learn and adapt to a changing environment. This is an extension to Sunrise which itself is an ensemble method which supports using SAC actors as the base learners.

Here you will find a introduction of the Sunrise algorithm and the ideas that would make up the DSunrise algorithm.

2 The Sunrise Algorithm

The Sunrise is a algorithm framework for ensemble learning. For simplicity we will assume Soft Actor Critic (SAC) are used as the base learners. Then Sunrise does three interesting things to make the ensemble works. These are:

- **Weighted bellman backups** - This is where the strength of the update to the actors and the critics is weighted by the standard deviation of the ensemble of the target critic estimates for a given state. This means that situation where the critic agree more will have a stronger update in line with the intuition that it is easier to agree on the truth (correct value).
- **Random initialization and training instances** - To increase diversity of the ensemble of base learners each base learner is randomly initialized and trained on a random subset of the replay buffer. The random subset of the replay buffer is achieved by having each transition given a mask that indicates which base learner can use it to train with.
- **UCB exploration** - To improve the exploration of the agent one can use the uncertainty of the ensemble to guide actions into unexplored areas. In practice this means at each time step choosing the action from the ensemble that maximizes both the expected reward and the standard deviation of the ensemble's value predictions.

Furthermore because of the nature of the ensemble it is possible that the ensemble can be parallelized which can result in a wall clock time comparable to single learner algorithms.

3 The Dsunrise Algorithm

The DSunrise algorithm is the same as the sunrise algorithm with the addition of a dynamic ensemble. That is on a regular interval (say every 10 episodes) the ensemble is check to see if any of the base learners can be removed. On removal of a base learner a new learner will be instantiated and added to the ensemble. Therefore the two important aspects of the DSunrise algorithm is the removal check and the instantiation of new base learners.

3.1 Removal Check

A removal check is a routine that is run on some regular interval (every step, episode, n episodes etc). As there is no managing critic the check only has the current base learners and the historic performance of these base learners to look at. From this information two classes of metrics can be created one is a performance check and the second is a diversity check.

The performance check can either look at historic performance and/or the predicted performance of the base learner. Historic performance could involve a type of time weighted average from the last n episodes. The predicted performance could be the average expected return from a sample from the replay buffer. One could either remove base learners that are below a certain threshold or simply remove the l poorest performing base learners.

The diversity check is trying to quantify how different the base learners are from each other. As the goal of an ensemble is that the base learners are different from each other removing base learners that happen to be too similar to each other will reduce the computational cost of the ensemble and provide opportunity for new base learners to be added to increase diversity. The diversity of a base learners can be understood by either how similar the critics are or how similar the proposed actions are from the polices. Furthermore as there is a unique optimal policy the ensemble of base learners should converge to this optimal policy and so the size of the ensemble will naturally decrease as training progresses.

In practice it is likely that a combination of performance metric and diversity metric would result in the best results. This could look something like

$$\text{performance}_i^{(t)} = \sum_{\tau=t-n}^{t} \omega_\tau \cdot R_i^{(\tau)} \quad \text{where} \quad \omega_\tau = \frac{\gamma^{t-\tau}}{\sum_{k=t-n}^{t} \gamma^{t-k}}$$

$$\text{diversity}_i^{(t)} = \frac{1}{|\mathcal{S}| \cdot (|\mathcal{E}| - 1)} \sum_{s \in \mathcal{S}} \sum_{\substack{j=1 \\ j \neq i}}^{|\mathcal{E}|} |\boldsymbol{a}_i(s) - \boldsymbol{a}_j(s)|_2$$

Where:

- $\text{performance}_i^{(t)}$: Performance metric for learner i at episode t
- $\text{diversity}_i^{(t)}$: Diversity metric for learner i at episode t
- $R_i^{(\tau)}$: Return of learner i in episode τ

- γ: Temporal discount factor ($0.9 \leq \gamma \leq 0.99$)
- n: Window size for performance evaluation
- $\mathcal{S}$: State sample from replay buffer ($|\mathcal{S}| = m$)
- $\mathcal{E}$: Current ensemble of learners
- $\boldsymbol{a}_i(s)$: Action vector of learner i in state s

The removal decision combines both metrics:

$$\text{remove}_i^{(t)} = \begin{cases} 1 & \text{if } \dfrac{\text{performance}_i^{(t)}}{\mu_P^{(t)}} < \theta_P \text{ and } \dfrac{\text{diversity}_i^{(t)}}{\mu_D^{(t)}} < \theta_D \\ 1 & \text{if } \dfrac{\text{performance}_i^{(t)}}{\mu_P^{(t)}} < \theta_{\text{crit}} \\ 0 & \text{otherwise} \end{cases}$$

With:

- $\mu_P^{(t)} = \frac{1}{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{E}|} \text{performance}_k^{(t)}$
- $\mu_D^{(t)} = \frac{1}{|\mathcal{E}|} \sum_{k=1}^{|\mathcal{E}|} \text{diversity}_k^{(t)}$
- θ_P: Performance threshold (≈ 0.7)
- θ_D: Diversity threshold (≈ 0.6)
- θ_{crit}: Critical performance threshold (≈ 0.5)

3.2 Instantiation of new base learners

When a base learner is removed from the ensemble a new base learner can be instantiated. There are many different ways to instantiate new models.

One way of instantiating a new base learner is to randomly instantiate a new base learner then train it up on some random subset of the replay buffer. This will add variation to it through the random initialization and the random subset of the replay buffer. However depending on the size of the experience subset this could be computationally similar to simply having the base learner as part of the ensemble since the beginning of the training.

Another way to instantiate a new base learner is for it to somehow be a variation of a current base learner. This could be done in a variety of ways which can broadly be separated into either cross over (involving multiple base learners) or mutation (involving a single base learner).

Once the instantiation of the new base learner is complete it could pass through a removal check to see if it is worth keeping in the ensemble. If it is not worth keeping then the ensemble size will naturally decrease. This naturally decreasing ensemble size is a feature that trends towards a single base learner which is computationally efficient (and potentially the optimal policy, depending on the guarantees of the removal check).

My exploration of new base learners instantiation will start with something like this.

Algorithm 1 Generate Replay Subset

Require: Replay memory $\mathcal{D}$, Reward threshold r_{top}, Error threshold e_{top}, Random sample ratio ρ_{random}

Initialize empty replay subset $\mathcal{D}_{\text{subset}} \leftarrow \emptyset$

Sort transitions in $\mathcal{D}$ by reward in descending order

Select top 50% transitions by reward:

$$\mathcal{D}_{\text{reward}} \leftarrow \text{Top } \lfloor 0.5 \cdot |\mathcal{D}| \rfloor \text{ transitions by reward}$$

Sort transitions in $\mathcal{D}$ by error in descending order

Select top 30% transitions by error:

$$\mathcal{D}_{\text{error}} \leftarrow \text{Top } \lfloor 0.3 \cdot |\mathcal{D}| \rfloor \text{ transitions by error}$$

Randomly sample 20% transitions from $\mathcal{D}$:

$$\mathcal{D}_{\text{random}} \leftarrow \text{Randomly sample } \lfloor 0.2 \cdot |\mathcal{D}| \rfloor \text{ transitions}$$

Combine subsets:

$$\mathcal{D}_{\text{subset}} \leftarrow \mathcal{D}_{\text{reward}} \cup \mathcal{D}_{\text{error}} \cup \mathcal{D}_{\text{random}}$$

return $\mathcal{D}_{\text{subset}}$

Algorithm 2 Instantiate New Base Learner

Require: Replay buffer $\mathcal{D}$, Ensemble $\mathcal{E}$, Mutation noise σ, Diversity threshold θ_D

Select a random base learner base $\in \mathcal{E}$

Generate mutation noise $\boldsymbol{\delta} \sim \mathcal{N}(0, \sigma^2)$

Instantiate new base learner:

$$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{base}} + \epsilon \cdot \boldsymbol{\delta}$$
$$\boldsymbol{\phi}_{\text{new}} \leftarrow \boldsymbol{\phi}_{\text{base}} + \epsilon \cdot \boldsymbol{\delta}$$

Call **Generate Replay Subset** to create training data:

$$\text{replay subset} \leftarrow \textbf{Generate Replay Subset}(\mathcal{D}, r_{\text{top}}, e_{\text{top}}, \rho_{\text{random}})$$

Train the new base learner on the replay subset

Perform diversity check:

$$\text{diversity}_{\text{new}} \leftarrow \frac{1}{|\mathcal{S}| \cdot (|\mathcal{E}| - 1)} \sum_{s \in \mathcal{S}} \sum_{\substack{j=1 \\ j \neq \text{new}}}^{|\mathcal{E}|} |\boldsymbol{a}_{\text{new}}(s) - \boldsymbol{a}_j(s)|_2$$

if $\text{diversity}_{\text{new}} < \theta_D$ **then**
    Discard new base learner
else
    Add new base learner to ensemble $\mathcal{E}$
end if

3.3 Basic algorithm

The starting point for the algorithm can be put together as follows:

Algorithm 3 Dynamic Sunrise Algorithm

Require: Replay buffer size N, Minibatch size K, discount factor γ, learning rates λ, λ_π and λ_Q, update frequency C, target smoothing coefficient τ, Updates to Data ratio UTD, environemt steps S, neural network function approximator Q, neural network function approximator π and initial entropy coefficient α, ensemble size E, Bellman weight temperature T,

Initialize replay memory $\mathcal{D} \leftarrow \emptyset$ with capacity N
Initialize policy π_ϕ with random weights ϕ
Initialize both action-value function Q_{θ_i} with random weights θ_i
Initialize $\bar{\theta}_i \leftarrow \theta_i$ for $i \in \{1, 2\}$
repeat
    for S steps **do**
        // UCB exploration alternatively randomly select a base learner to be used for each episode.
        Collect E action samples $\boldsymbol{a}_t^{(i)} = \pi_\phi(s_t)$ for $i \in \{1, 2, \ldots, E\}$
        select action: $a_t \sim \pi_\phi(a_t|s_t)$
        Choose the action that maximizes the UCB exploration term:
        $a_t = \arg\max_a \left(Q_{\theta_1}(s_t, a) + \alpha \cdot \sigma(Q_{\theta_2}(s_t, a))\right)$
        Observe reward r_t and next state s_{t+1}
        Sample bootstrap masks $m_t = \{m_{t,i} \sim \text{Bernoulli}(0.5) \text{ for } i \in \{1, ..., E\}\}$
        Store transition $(s_t, a_t, r_t, s_{t+1}, m_t)$ in $\mathcal{D}$
    end for
    if Size of $\mathcal{D}$ is less than 10,000 **then**
        Continue to next iteration without training of removal checks
    end if
    for UTD times **do**
        Sample a batch B of size K from $\mathcal{D}$
        for each agent i **do**
            Use Mask $m_{t,i}$ to select transitions from B for agent i
            Update Q-functions to minimize $(\text{sigmoid}(-\bar{Q}_{\text{std}}(s_{t+1}, a_{t+1}) * T) + 0.5)(Q_{\theta_i}(s_t, a_t) - r_t - \gamma \bar{V}(s_{t+1}))$
            Update policy to minimize $\mathbb{E}_{a_t \sim \pi_\phi}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right]$
            Update entropy coefficient: $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
            Update target Q-networks: $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
        end for
    end for
    Perform removal check on the ensemble of base learners
    for Removed base learners **do**
        Instantiate new base learner using the instantiation algorithm
        Update T to reflect the new ensemble size
    end for
until stopping criteria is met

Note that this algorithm does not take into account the various extra bits of informa-

tion that will need to be stored for the proposed removal check and the instantiation of new base learners.

4 Conclusion

The sunrise algorithm is a simple but extensible framework for ensemble learnings. By making the ensemble dynamic it allows the algorithm to explore with greater speed and still return a single base learner.

The dynamic nature does not interfere with the ability of the original algorithm to be parallelized to speed up wall clock time. Furthermore it gives the ability for the ensemble to adapt to a changing environment by slowly pruning the less effective base learners and replacing them with new ones.