CS 499 Milestone Three: Algorithms and Data Structures Enhancement Narrative

Joseph Klenk
CS 499 Computer Science Capstone
Category: Algorithms and Data Structures

Artifact Description

The improved artifact is a Pirate Intelligent Agent from CS 370, initially adopting standard deep Q-learning for pathfinding through a maze. Optimal pathfinding is leanred through reinforcement learning with the use of epsilon-greedy exploration and experience replay to navigate to treasure avoiding obstacles.

Justification for Inclusion

I chose this artifact because it illustrates mastery of data structures and advanced algorithms with measurable performance gains. The initial implementation employed the use of simple random sampling for experience replay, offering a clear opportunity to leverage advanced algorithmic optimizations based on industry standards.

Key Enhancements

Priority Queue Implementation: Replaced the initial list-based random sampling with a heap-based priority queue via the heapq module in Python. This gives O(log n) insertion/extraction operations against O(n) for keeping sorted order.

Temporal Difference (TD) Error Calculation: Utilized implemented algorithms for calculating the learning value of each experience and ranked experiences with the top prediction errors. Through this mathematical method, the agent learns from the most valuable experiences first.

Intelligent Sampling Algorithm: Created probability distribution algorithms for sampling experience based on importance, not at random, marking a profound leap toward value-based optimization of learning.

Technical Implementation

The improved GameExperience class includes original and priority-based methods using the use_priority parameter. When used, it computes TD-error for each episode with the formula: |target_Q - current_Q|, with larger errors getting greater priority. Episodes are stored in the heap as (-priority, index, episode) tuples, with negative priorities for a max-heap for optimal sampling.

Course Outcomes Achievement

Developed and tested computing solutions based upon algorithmic principles. The implementation of the priority queue shows examination of known algorithms, detection of the inefficiencies, and design of better solutions with explicit trade-offs in performance.

Applied advanced techniques developed from reinforcement learning research, in particular, priority experience replay methods from sophisticated AI systems, demonstrating sound algorithmic solutions.

Results and Impact

Testing demonstrates the enhanced algorithm achieves **28.8% faster convergence** (156 vs 219 epochs to reach 100% win rate), validating the theoretical improvements. The implementation maintains all original functionality while adding sophisticated priority-based sampling, resulting in more efficient training and reduced computational requirements.

Reflection

This project improved my knowledge of the interaction between data structures and algorithm performance. Working with the TD-error computations educated me about mathematical concepts that underpin machine learning optimizations. The greatest challenge was handling priority queue operations while keeping the memory constraints in place, addressed with a hybrid method that automatically eliminates low-priority experiences.

The project enhanced my skills in applying research-based algorithms and showed the impact of appropriate data structure selection in converting a working system into an optimal, professional-level solution. The project is the strongest evidence of my preparedness to address complex algorithmic problems in AI and machine learning contexts.