MACHINE LEARNING HACKATHON
ANALYSIS REPORT
HANGMAN

MEMBERS: **DATE: 03/11/2025**

1) **PES2UG23CS005**
2) **PES2UG23CS015**
3) **PES2UG23CS030**
4) **PES2UG23CS033**

1. Key Observations

1.1 Most Challenging Parts

Challenge 1: The Sparse Reward Problem The most significant challenge was the extreme sparseness of positive rewards. With only 26.80% wins, the agent experienced negative feedback 73.20% of the time. This creates a severe learning difficulty where the agent must discover successful strategies through predominantly negative experiences.

Challenge 2: High-Dimensional Action Space with Context Dependency The action space (26 letters) combined with varying word lengths (1-24 characters) and partial reveal patterns creates an enormous state-action space. Each decision is highly context-dependent - the optimal letter choice changes dramatically based on:

- Current pattern (revealed/hidden letters)
- Word length
- Previously guessed letters
- Remaining lives

Challenge 3: Overfitting vs. Generalization Trade-off The training corpus contains 50,000 words with specific statistical patterns. The agent must learn these patterns without overfitting to training data idiosyncrasies. The 2,000-episode training regime was explicitly designed to combat overfitting, but this conservative approach may have limited learning potential.

Challenge 4: Non-Stationary Learning Environment As the RL agent updates its letter biases, the policy changes, which affects the distribution of states encountered. This creates a moving target for learning, making convergence difficult.

1.2 Key Insights Gained

Insight 1: Statistical Priors are Critical The HMM component provides essential statistical priors that prevent the RL agent from random exploration. Without these priors, the 26-letter action space would require exponentially more episodes to learn effective strategies. The HMM provides:

- Letter frequency information: 'e' (10.37%), 'a' (8.87%), 'i' (8.86%)
- Position-specific probabilities: 'e' ends 18.61% of words
- Contextual patterns: 'er', 'in', 'ti' are common bigrams
-

Insight 2: Accuracy ≠ Win Rate The agent achieves 50.61% guess accuracy but only 26.80% win rate. This reveals that:

- Correct guesses don't guarantee wins
- The 6-life constraint is severe
- Early mistakes are disproportionately costly
- Word completion requires near-perfect letter sequences in difficult cases

Insight 3: Epsilon Decay Behavior The epsilon decayed from 0.20 to 0.0499 (effective minimum) very quickly (by episode ~300). This rapid convergence to exploitation suggests:

- The agent found a locally optimal strategy early
- Further exploration was abandoned prematurely
- The epsilon decay rate (0.995) was too aggressive for this problem

Insight 4: Performance Plateau Examining the training metrics shows the win rate plateauing around 24-26% after ~200 episodes, with no significant improvement through 2,000 episodes. This suggests:

- The current architecture has hit its learning ceiling
- More episodes wouldn't help without architectural changes
- The combination of HMM + simple Q-learning may be insufficient

2. Strategy Analysis

2.1 HMM Design Choices

Multi-Level Probability Model:
The HMM implements a hierarchical probability system with weighted components:

```
python
weights = {
   'global': 0.20,      # Overall letter frequency
   'length': 0.25,      # Length-specific patterns
   'position': 0.25,    # Positional probabilities
   'context': 0.30      # Conditional probabilities (highest)
}
```

Rationale:

1. Global Frequency (20%): Baseline letter importance, ensures common letters prioritized
2. Length-Specific (25%): Captures that 9-letter words have different patterns than 3-letter words
3. Position-Based (25%): Uses fixed positions (first: 'p' 10.40%, last: 'e' 18.61%) and 10 relative bins
4. Context-Based (30%): Leverages revealed letters to predict adjacent unknowns using bigram frequencies

The context component receives the highest weight because conditional probabilities (e.g., 'q' → 'u') provide the strongest predictions when letters are already revealed.

Vowel-Consonant Heuristic: Adaptively boosts vowel probabilities by 1.5× when consonants dominate (more than vowels + 2), recognizing English words typically have ~39% vowels.

2.2 RL State Design

Discretized State Features:
python
state = (length_bucket, progress_bucket, lives_bucket)
# 5 × 5 × 3 = 75 total discrete states

- Length buckets: [0-2], [3-5], [6-8], [9-11], [12+] letters
- Progress buckets: [0-25%], [26-50%], [51-75%], [76-99%], [100%] revealed
- Lives buckets: [0-1], [2-3], [4-6] lives remaining

Design Philosophy: Discretization enables generalization across similar situations instead of treating each exact state uniquely. This trades detailed state information for faster learning and better generalization across the 75-state space.

2.3 Reward Design

Structure:

- Correct guess: +10 per revealed letter + 2 (range: +12 to +52)
- Wrong guess: -10 - (6 - lives_remaining) × 3 (range: -13 to -28, escalating)
- Repeated guess: -10
- Win: +100 | Loss: -100

Key Principles:

1. Asymmetric rewards encourage conservative, high-confidence guesses
2. Escalating penalties create urgency as lives decrease
3. Multi-letter bonus rewards pattern recognition
4. Dominant terminal rewards shape long-term strategy toward winning, not just correct guessing

2.4 Integration Philosophy

The HMM + RL combination implements hierarchical decision-making:

- HMM (Strategic Layer): Provides domain knowledge and statistical priors
- RL (Tactical Layer): Fine-tunes decisions using a learned letter_bias array
python
adjusted_prob = hmm_prob × (1.0 + bias × 0.2)
This allows the RL agent to boost/penalize specific letters based on experience while maintaining the HMM's statistical foundation.

3. Exploration vs. Exploitation Trade-off

3.1 Implemented Strategy: ε-Greedy with Decay

Configuration:

- Initial ε = 0.20 (20% exploration)
- Decay rate = 0.995 per episode
- Minimum ε = 0.05 (5% floor)

Trajectory: Reached minimum by episode 300 (~85% of training in full exploitation mode)

3.2 Critical Analysis

What Worked:

- HMM-guided exploration prevents wasting exploration on obviously poor choices
- Letter bias array successfully captures simple preferences across states

What Didn't Work:

1. Too Aggressive Decay: Performance plateaued at ~25% win rate by episode 200 with no improvement through episode 2,000
2. Premature Convergence: Agent locked into suboptimal policy too early
3. Insufficient Exploration: With 26 actions and high complexity, 20% initial epsilon was too conservative

The Core Problem: Even during "exploration," the agent uses probability-weighted sampling from HMM probs rather than uniform random selection. This is safe but may miss discovering unconventional strategies. Combined with rapid epsilon decay, the agent never adequately explores the full action space.

4. Future Improvements

1. Adaptive epsilon decay
2. Experience replay buffer
3. Enhanced reward shaping

Rationale: These require minimal code changes but address the core exploration-exploitation problem.
Expected Combined Impact: +20-30% win rate improvement (to ~47-57% win rate)

4. Performance Summary

Current Results

- Win Rate: 26.80% (536/2,000 games)
- Guess Accuracy: 50.61%
- Avg Wrong Guesses/Game: 4.66
- Final Score: -46,099

Performance Gap

- Human Expert: 70-85% win rate
- Current Agent: 26.80% win rate
- Gap: Agent performs at ~1/3 of human expert level

Primary Weaknesses

1. Consonant blindness: Struggles with less common consonants ('g', 'h', 'm', 'p', 'f')
2. Middle-letter problem: Can't fill patterns like "_er___e" → "perfume"
3. No look-ahead: Makes locally optimal guesses without planning
4. Premature convergence: Stopped learning after 200 episodes

Conclusion

The current system demonstrates a solid foundation combining statistical priors (HMM) with adaptive learning (RL). However, the aggressive exploration-exploitation trade-off and simple state representation limit performance to 26.80% win rate. The recommended improvements—particularly adaptive epsilon, experience replay, and DQN architecture—address these core limitations and could realistically achieve 50-70% win rate, approaching human-level performance.

&&&&&&&&&&&&THANK YOU&&&&&&&&&&&&