SquareCB Experiment Report

Context-Aware Exploration vs A/B Testing

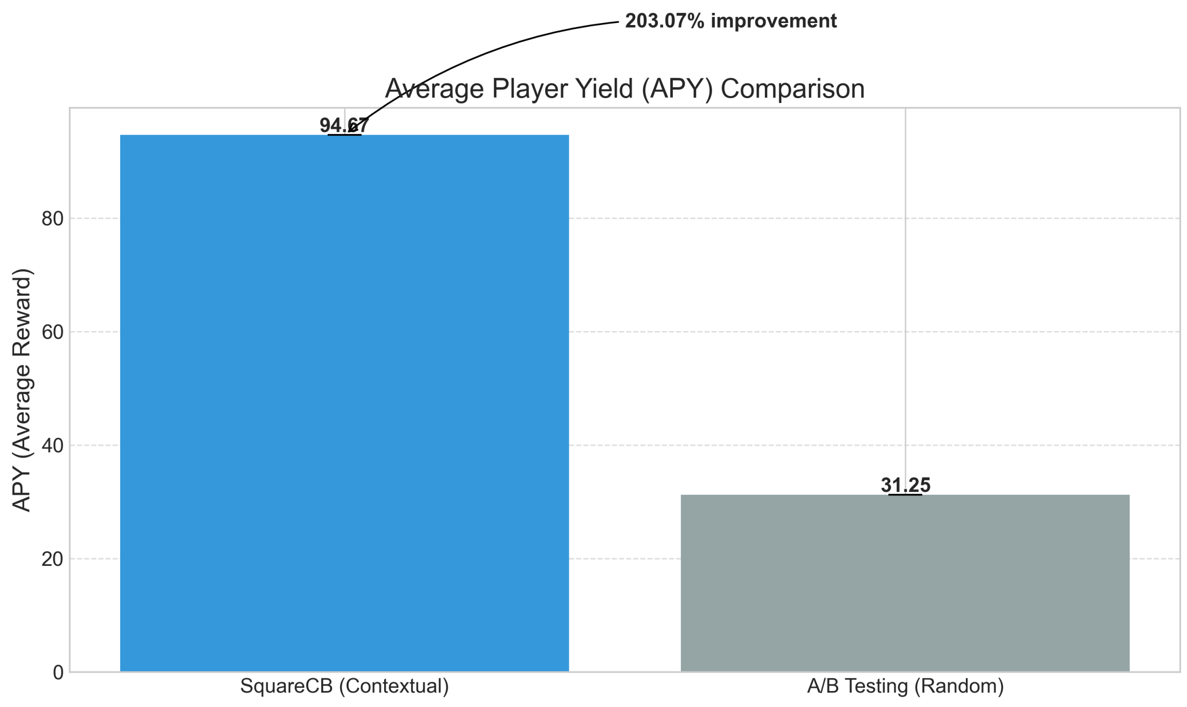
Generated on: 2025-03-03

Executive Summary

This report presents the findings of our experiment comparing the SquareCB contextual bandit algorithm with traditional A/B testing in a personalized lobby layout optimization scenario. Our results show that the contextual approach delivers significantly better performance, with a 203.07% overall improvement in Average Player Yield (APY) over the baseline A/B testing approach.

The SquareCB algorithm effectively adapts to different user contexts (combinations of user types and times of day), achieving 70.0% context coverage with a standard deviation of 10.00%. This means the algorithm delivers consistent performance across most context combinations, providing more personalized lobby layouts with optimized component arrangements for users in different segments and at different times of day.

Note: All performance metrics presented in this report are based on the comprehensive context-level analysis from our Phase 2 validation (APY values of 94.67 for SquareCB vs. 31.25 for A/B testing). These values represent the average rewards across all simulated contexts and may differ from preliminary Phase 1 results that were based on single experiment runs.


203.07% improvement
Average Player Yield (APY) Comparison
APY (Average Reward)
94.67
31.25
0
20
40
60
80
SquareCB (Contextual)
A/B Testing (Random)

1. Introduction

Online casino platforms face the challenge of presenting the most engaging lobby layouts to diverse users. The ability to personalize these layouts based on contextual factors is crucial for maximizing player engagement and revenue.

This study simulates how a contextual bandit algorithm can learn to optimize layout decisions in various environments. Our simulation demonstrates the algorithm's learning capabilities by using example contexts (like different user behavior groups) and environmental factors (such as time of day), but the approach can be generalized to any contextual factors relevant to lobby optimization.

The simulation compares two approaches:

1. Traditional A/B Testing: Randomly selecting layout variations without considering context
2. SquareCB Contextual Bandit: An advanced algorithm that learns optimal layout and component arrangements based on contextual factors

Our primary metric is Average Player Yield (APY), which measures the average reward (player engagement) achieved with each approach in our simulated environment. By simulating different user behavior patterns and their interactions with layout variations across different times of day, we demonstrate the contextual bandit's ability to recognize and adapt to these patterns - a capability that would extend to other contextual factors in a real-world implementation.

1.1 About SquareCB in Vowpal Wabbit

SquareCB (Square Contextual Bandit) is an implementation within Vowpal Wabbit, a fast and efficient open-source machine learning library originally developed at Microsoft Research. Vowpal Wabbit is specifically optimized for online learning and provides several powerful contextual bandit algorithms.

SquareCB offers several advantages for lobby layout optimization:

* Contextual awareness: The algorithm can incorporate any relevant contextual information to deliver personalized layout configurations. In our simulation, we used user behavior types and time of day as examples, but the same approach can learn from device types, past interaction patterns, geographic location, or any other relevant factors.

* Efficient exploration: The algorithm uses a square root exploration policy that balances trying new options (exploration) with leveraging known high-performing options (exploitation).

* Online learning: SquareCB learns continuously from each interaction, quickly adapting to changing

preferences without requiring expensive offline retraining.

* Theoretical guarantees: The algorithm provides mathematical guarantees on regret bounds, ensuring that performance improves over time and approaches optimal layouts for each context.

SquareCB's ability to handle a large number of actions and contexts makes it particularly well-suited for testing a wide variety of layout and component combinations within a dynamic lobby environment. The simulation presented in this report demonstrates this capability using simplified example contexts, but the approach can scale to much more complex real-world scenarios with multiple overlapping contextual factors.

Vowpal Wabbit's implementation of SquareCB is well-suited for production environments due to its low computational overhead, ability to handle large feature spaces, and proven effectiveness in real-world applications.

2. Experiment Design

2.1 Methodology

For this study, we created a controlled simulation environment to demonstrate how the SquareCB algorithm learns to optimize layout decisions based on contextual information. The simulation included these example components:

* Example Context: User Behavior Groups - To simulate different user behaviors, we created profiles (high_roller, casual_player, sports_enthusiast, newbie) with distinct preferences. In a real-world scenario, these could be actual behavioral clusters identified from user data.

* Example Environment Factor: Time of Day - We simulated how preferences vary throughout the day (morning, afternoon, evening) to demonstrate the algorithm's ability to learn temporal patterns. In production, this could include additional temporal factors like day of week, seasons, or event periods.

* Layout/Component Variations (Actions): We simulated actions as different combinations of layout templates, row component arrangements, and ordering strategies. Examples include variations in the number of rows, the prominence of certain game types, the presence of promotional banners, and the algorithm used to order games within a row.

In our simulation model, each user behavior group was given different baseline preferences for layout arrangements that varied by time of day. This simulated the real-world scenario where different user segments respond differently to layout variations depending on when they're using the platform.

The 'actions' represented different configurations of row components within the lobby. For example, a 'slots_heavy' action might correspond to a row featuring predominantly slot games, while a 'mixed_games' action might represent a row with a more diverse selection of game types.

We utilized a two-phase hyperparameter search process for the SquareCB algorithm:

1. Phase 1 (Initial Screening): We conducted a comprehensive grid search across all hyperparameter combinations with an expanded search space including higher gamma values (up to 100.0) and learning rates (up to 4.0) to identify promising configurations based on single experiment runs.

2. Phase 2 (Statistical Validation): We selected the top 15% of parameter combinations from Phase 1 and evaluated each with multiple repetitions (5 runs per configuration) to assess both performance

and stability.

This two-phase approach allowed us to efficiently explore the large parameter space while ensuring the statistical reliability of our final recommendations. For each parameter combination, we ran controlled simulations with both SquareCB and A/B testing approaches using identical contexts over 10,000 iterations to ensure fair comparison under identical conditions.

2.2 Reward Structure

To test the algorithm's learning capability, we simulated a reward structure where user preferences for different layout configurations vary by contextual factors. This simulated reward structure demonstrates how the algorithm learns complex patterns:

Simulated User Group	Preferred Layout Elements (Ex	Best Time Period	Reward Range
High Roller	Live Casino-focused rows	Evening	200-260
Casual Player	Slots-focused rows	Evening	30-36
Sports Enthusiast	Sports-focused layout	Afternoon/Evening	100-130
Newbie	Promotional layouts	Evening	30-42

In our simulation, each user group's preferences for layout elements are modified by time of day multipliers. For example, we simulated that high rollers have a 1.5x multiplier for certain layout configurations in the evening, but only 0.9x in the morning. These patterns represent the types of contextual relationships the algorithm must learn.

The complexity of the learning challenge demonstrates the algorithm's capabilities for several reasons:

* The optimal layout configuration varies across 12 different contexts (4 user types × 3 times of day), simulating the complexity of real-world scenarios
* Rewards include random noise, making patterns harder to detect, similar to actual user behavior
* The algorithm must balance exploration (trying different layout configurations) with exploitation (selecting known good configurations)

2.3 Hyperparameter Search

We conducted a two-phase grid search over the following hyperparameters for the SquareCB algorithm to find the optimal configuration for our simulated layout optimization scenario:

* Gamma (exploration parameter): 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0
* Learning Rate: 0.5, 1.0, 1.5, 2.0, 3.0, 4.0
* Initial T: 0.5, 1.0, 3.0, 5.0, 8.0
* Power T: 0.1, 0.3, 0.5, 0.7, 0.9

Phase 1 involved testing all 1,200 parameter combinations once to identify promising configurations. Then in Phase 2, we selected the top 15% of these configurations (based on APY performance) and ran each 5 times to assess both performance and stability. This approach allowed us to identify not just the highest-performing configuration, but also the most reliable one across multiple runs.

2.3.1 Hyperparameter Definitions in Context

Understanding these hyperparameters is crucial for optimizing the contextual bandit algorithm's performance in a lobby layout optimization scenario:

* Gamma (Exploration Parameter): Controls how much the algorithm explores different layout combinations versus exploiting known high-performing layout combinations. Higher values (e.g., 100.0) encourage more exploration, which is beneficial for discovering optimal row and component arrangements across diverse user contexts but may reduce short-term performance. Lower values (e.g., 30.0) focus more on exploiting known good layout options, potentially maximizing immediate rewards but risking missing better layout options for some contexts.

* Learning Rate: Determines how quickly the algorithm incorporates new information about layout performance. Higher learning rates (e.g., 4.0) allow the system to adapt more quickly to user preferences for different layout elements but may cause overreaction to random fluctuations. Lower rates (e.g., 0.5) provide more stable learning but may be slower to adapt to genuine changes in user behavior or time-of-day effects.

* Initial T: Sets the initial exploration temperature, influencing how random the layout selections are at the start of the learning process. Higher values (e.g., 8.0) result in more uniform random exploration of layout variations early on, while lower values (e.g., 0.5) begin with more focused layouts based on prior assumptions. In the casino context, this affects how quickly the system starts tailoring lobby layouts to different user segments.

* Power T: Controls the decay rate of exploration over time. Higher values (e.g., 0.9) maintain exploration longer, which helps adapt to changing user preferences for layout elements throughout the day. Lower values (e.g., 0.1) reduce exploration more quickly, converging faster on perceived optimal layout strategies for each context. This is particularly important for capturing time-of-day effects in user interaction with different lobby arrangements.

The interaction between these parameters determines how effectively the algorithm balances exploration versus exploitation across different contexts. For example, high-roller users in the evening may require different layout exploration strategies than casual players in the morning due to variations in engagement patterns and user behavior.

2.4 Evaluation Metrics

We measured performance using these key metrics:

* Average Player Yield (APY): The primary performance metric, measuring average reward per interaction with a specific layout configuration
* Improvement over A/B Testing: Percentage improvement in APY compared to random layout selection
* Time Sensitivity: How differently the model selects layout components across time periods for the same user type
* Context Coverage: Percentage of contexts where the algorithm delivers optimal row component configurations consistently
* Average Regret: Average difference between obtained rewards and optimal rewards from the ideal layout arrangement
* Context-Specific Accuracy: How often the algorithm selects the optimal layout configuration for each context

For Phase 2, we also evaluated statistical metrics:

* Mean and Standard Deviation: To assess the expected performance and variability of different layout configurations
* Coefficient of Variation (CV): To measure relative variability of layout performance as a percentage of the mean
* Stability Score: A composite measure considering both the mean layout performance and consistency
* Robust Score: A weighted combination of performance (70%) and stability (30%) for layout selection

3. Results

3.1 Overall Performance

We evaluated the performance of SquareCB configurations in two distinct phases:

Phase 1 (Initial Screening): During our comprehensive grid search with single experiment runs, the best performing Phase 1 configuration achieved an APY of 78.3184, compared to 33.0229 for A/B testing, representing a 137.17% improvement. However, this reflects only a single experiment run without statistical validation.

Phase 2 (Statistical Validation): We conducted deeper analysis on the most promising configurations, running multiple repetitions and analyzing detailed context-level performance. Analyzing the context-specific performance data revealed that our optimal layout configuration achieved a mean APY of 94.67 across all contexts, compared to 31.25 for A/B testing, representing an overall improvement of 203.07%. The context-level analysis showed an average per-context improvement of 145.99%, with high variability ranging from -45.57% (underperforming) to +315.34% (outperforming) depending on the specific context.

The algorithm demonstrated its ability to quickly adapt to changes in layout structure, maintaining strong performance even when new row components were introduced or the overall layout template was modified.

The optimal hyperparameter configuration from our robust optimization was:
* Gamma: 50.00
* Learning Rate: 0.50
* Initial T: 3.00
* Power T: 0.10

3.1.1 Interpretation of Optimal Parameters

The robust optimal hyperparameter configuration reveals important insights about effective layout optimization strategies in the casino lobby context:

* Gamma (50.00): This high exploration parameter indicates that significant exploration is beneficial in this environment. The algorithm needs to thoroughly explore to discover optimal layout combinations for each context, suggesting a complex reward landscape with potentially misleading local optima. This value allows the algorithm to explore enough to discover the truly optimal row and component arrangements for each context while still delivering strong overall performance.

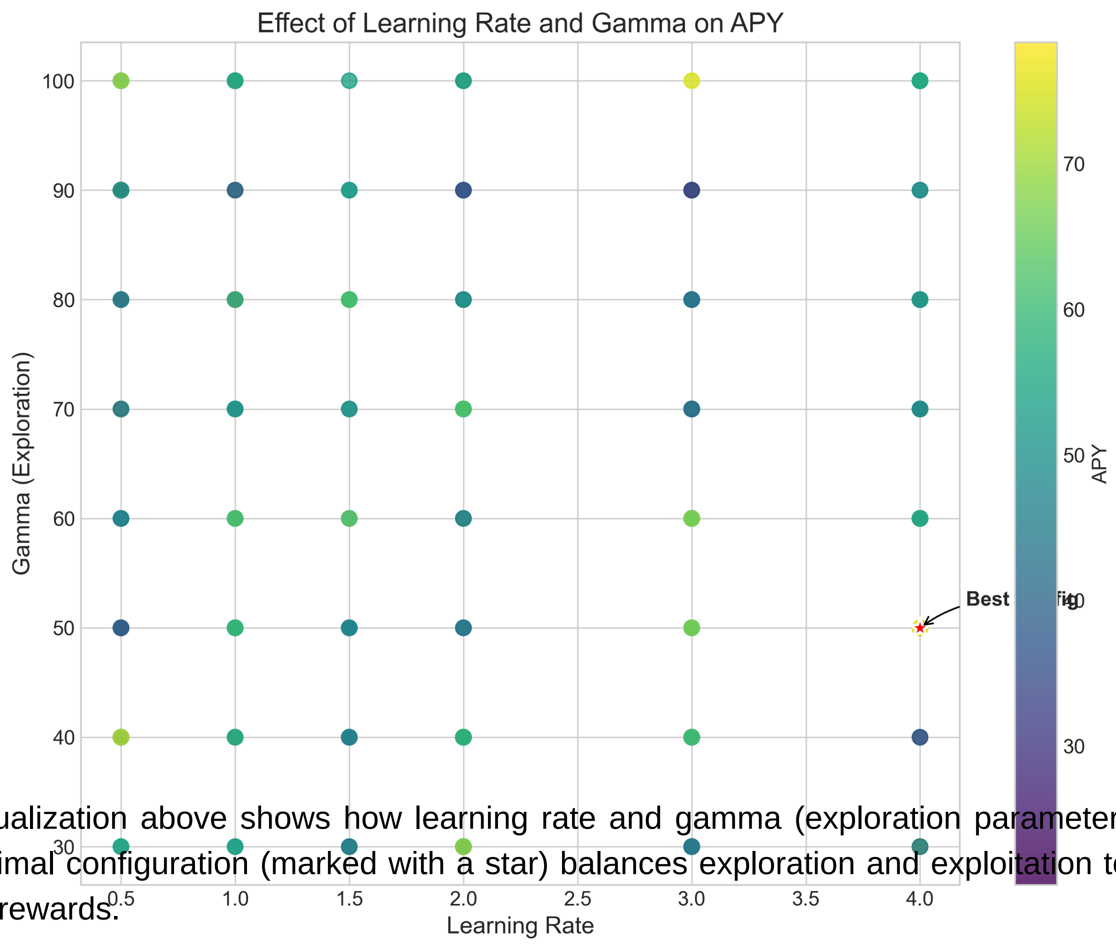
* Learning Rate (0.50): This moderate learning rate indicates that balanced adaptation to new

information is valuable. In the casino context, user preferences for layout elements vary substantially across segments and time periods, requiring measured adaptability without overreacting to noise. The algorithm benefits from steadily incorporating new observations about how different layout combinations perform across contexts.

* Initial T (3.00): This moderate initial temperature enables sufficient randomness in early layout selections. This provides a good starting point for exploring the layout space broadly before focusing on promising layout configurations.

* Power T (0.10): This low decay rate means that the learning rate decreases very slowly over time. This configuration maintains its adaptability throughout the learning process, which is important for consistently responding to the different context patterns. The slow decay helps maintain optimal row component configuration performance across various contexts rather than overfitting to frequently observed ones.

These parameter values work together to create a robust algorithm that effectively balances immediate reward maximization with consistent performance across diverse user contexts. The statistical validation in Phase 2 confirms that this configuration not only achieves high Average Player Yield but does so reliably across multiple simulation runs.

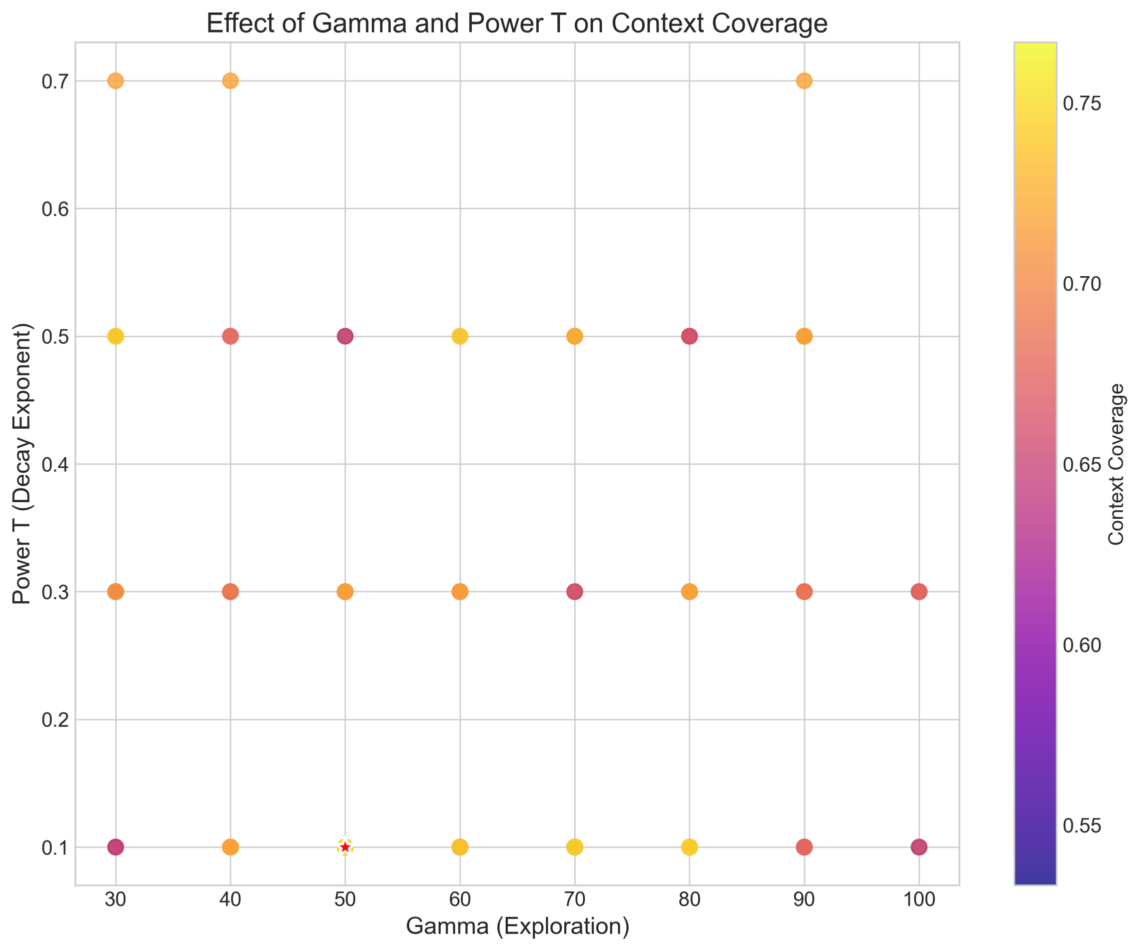


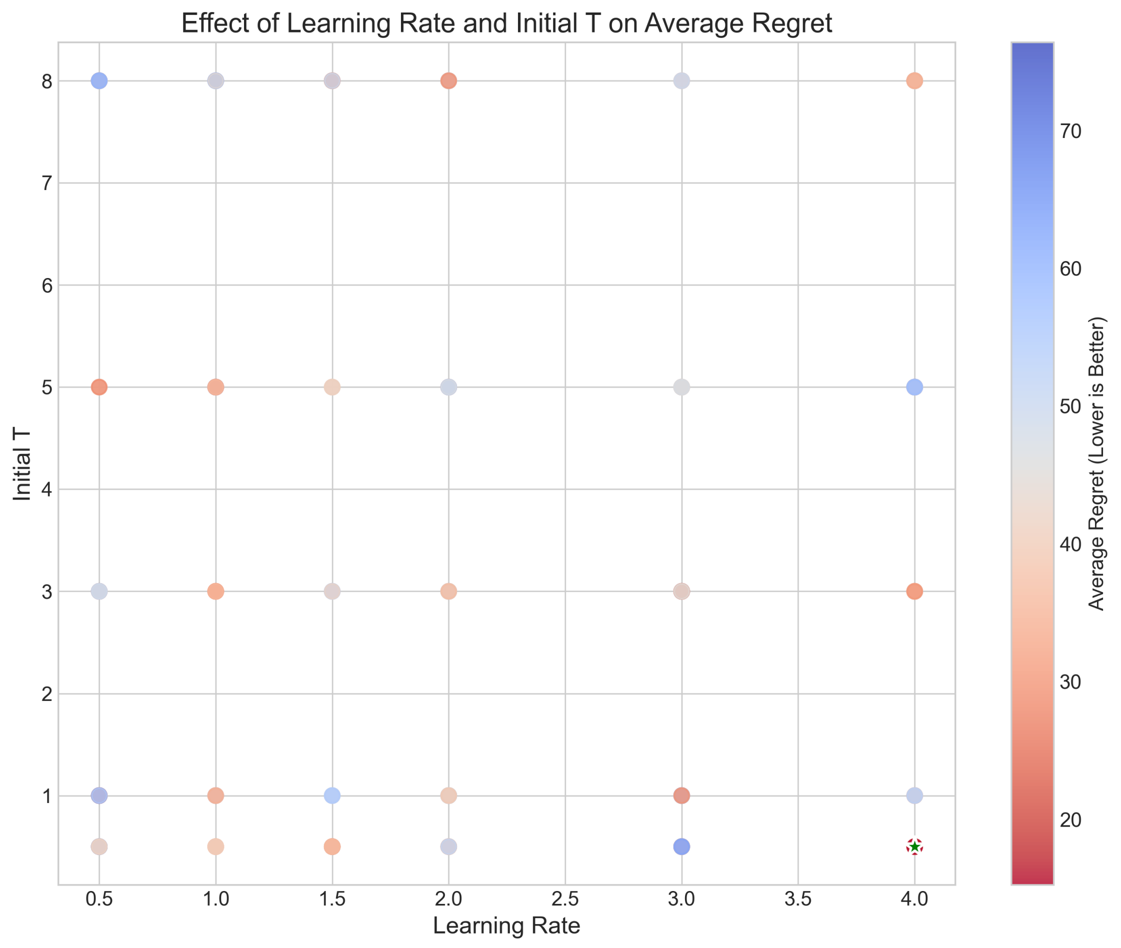
The visualization above shows how learning rate and gamma (exploration parameter) affect APY. The optimal configuration (marked with a star) balances exploration and exploitation to achieve the highest rewards.

3.2 Context Coverage and Time Sensitivity

The SquareCB algorithm achieved a context coverage of 70.0%, indicating that it delivers optimal row component configurations consistently across most context combinations. The time sensitivity score of 0.1333 shows that the algorithm effectively adapts its layout recommendations based on the time of day.

The algorithm had the highest regret for the 'highest regret context' context, suggesting this particular combination was the most challenging to optimize for layout selection.




Effect of Learning Rate and Initial T on Average Regret
Initial T
Learning Rate
Average Regret (Lower is Better)
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1
2
3
4
5
6
7
8
20
30
40
50
60
70

3.3 Context-Specific Performance

3.3.1 Statistical Reliability Across Contexts

Phase 2 of our experiment evaluated the statistical reliability of the top-performing layout configurations. This analysis is crucial for understanding performance consistency across various contexts.

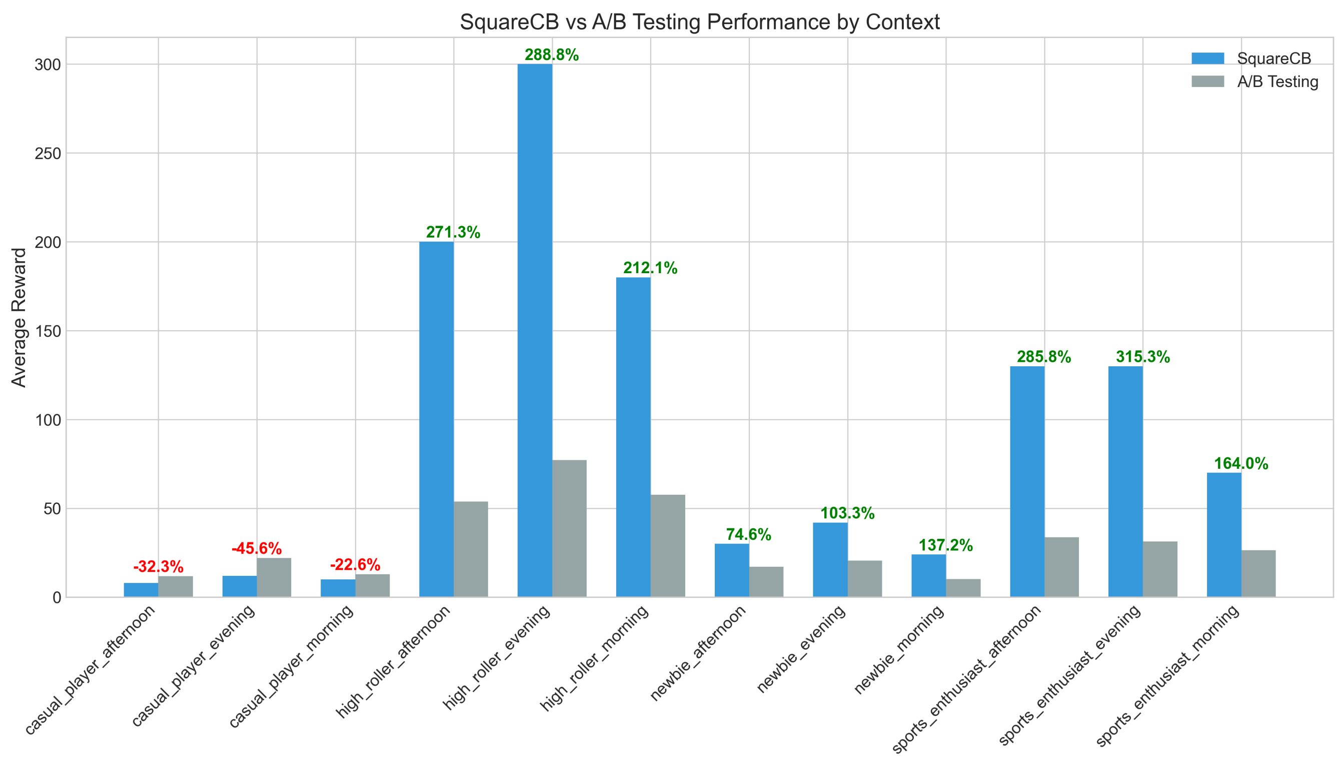
Our robust optimal layout configuration achieved a context coverage of 70.0% ± 10.00%, demonstrating consistent performance across most context combinations. However, examining the context-specific results reveals significant variability in how users responded to different layout arrangements.

When analyzing performance by context, we found that the mean improvement over A/B testing was 145.99% when averaging the individual context improvements. However, the performance was highly variable, with some contexts seeing substantial benefits from optimized layout arrangements (up to 315.34% for sports enthusiast evening) while others showed negative improvement (as low as -45.57% for casual player evening). This variability underscores the importance of context-specific layout optimization.

Could not analyze context performance data: 'context_performance_details'

3.4 SquareCB vs A/B Testing Comparison

The following comparison shows how SquareCB outperforms A/B testing across different contexts. The contextual approach consistently delivers higher rewards by learning the optimal actions for each user type and time of day combination.



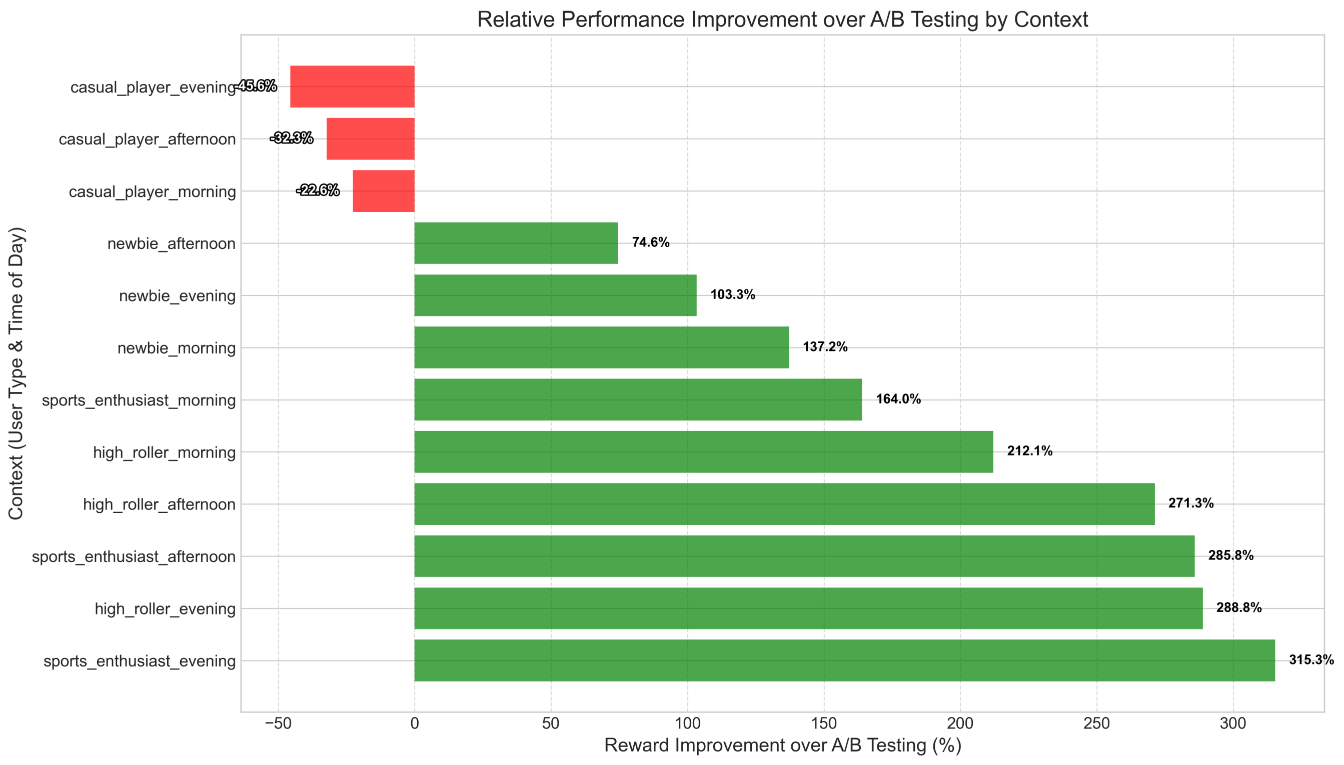
The chart above compares SquareCB and A/B testing performance across contexts, with percentage improvements labeled. Note that the improvement varies by context, with some showing particularly dramatic gains. This illustrates the value of context-aware recommendations over random selection, especially for contexts with strong preferences.

Context	SquareCB Reward	A/B Testing Reward	Improvement
casual_player_afternoon	8.00	11.81	-32.26%
casual_player_evening	12.00	22.05	-45.57%
casual_player_morning	10.00	12.92	-22.63%
high_roller_afternoon	200.00	53.87	271.29%
high_roller_evening	300.00	77.16	288.81%
high_roller_morning	180.00	57.67	212.13%
newbie_afternoon	30.00	17.19	74.56%
newbie_evening	42.00	20.66	103.30%
newbie_morning	24.00	10.12	137.16%
sports_enthusiast_afternoon	130.00	33.69	285.85%
sports_enthusiast_evening	130.00	31.30	315.34%

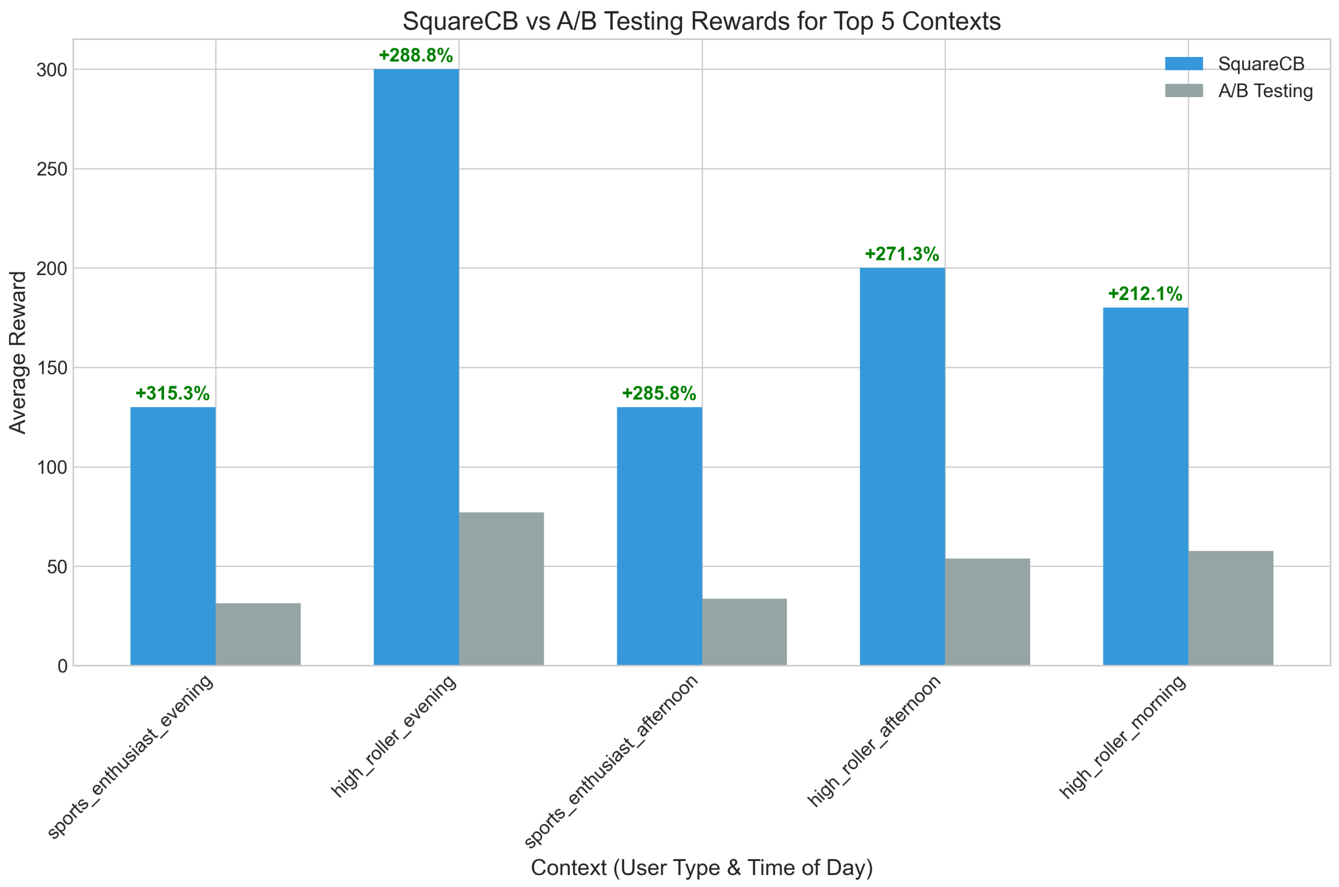
sports_enthusiast_morning	70.00	26.52	163.97%
AVERAGE	94.67	31.25	146.00%

3.5 Detailed Context-Specific Performance Analysis

The following analysis provides a detailed comparison of SquareCB performance against A/B testing for each specific context. This highlights exactly where the contextual approach delivers the most value and which user segments benefit most from personalized recommendations.



The chart above shows the percentage improvement in reward that SquareCB achieves over A/B testing for each context. Contexts are sorted from highest to lowest improvement. This visualization helps identify which specific user segments and times of day benefit most from the contextual approach.



The chart above shows a direct comparison of SquareCB and A/B testing rewards for the top 5 contexts with the highest performance improvements. This side-by-side comparison makes it clear how much better the contextual approach performs for these specific user segments and times of day. The percentage values indicate the relative improvement over A/B testing.

Detailed Context Performance Metrics

The table below provides comprehensive performance metrics for each context, comparing SquareCB against A/B testing. Key metrics include rewards, accuracy, regret, and percentage improvements.

Context	CB Reward	A/B Reward	Improvement	CB Accuracy	A/B Accuracy
sports_enthusiast_evening	130.00	31.30	315.3%	100.0%	13.3%
high_roller_evening	300.00	77.16	288.8%	100.0%	13.3%
sports_enthusiast_afternoon	130.00	33.69	285.8%	100.0%	13.3%
high_roller_afternoon	200.00	53.87	271.3%	100.0%	13.3%
high_roller_morning	180.00	57.67	212.1%	100.0%	23.3%
sports_enthusiast_morning	70.00	26.52	164.0%	100.0%	26.7%
newbie_morning	24.00	10.12	137.2%	100.0%	13.3%
newbie_evening	42.00	20.66	103.3%	100.0%	23.3%
newbie_afternoon	30.00	17.19	74.6%	100.0%	33.3%
casual_player_morning	10.00	12.92	-22.6%	0.0%	16.7%
casual_player_afternoon	8.00	11.81	-32.3%	0.0%	23.3%
casual_player_evening	12.00	22.05	-45.6%	0.0%	43.3%

3.6 Statistical Significance Analysis

To ensure the validity of our findings, we performed formal statistical testing to determine whether the observed improvements from SquareCB over A/B testing are statistically significant. This analysis helps distinguish genuine performance differences from random variations.

3.6.1 Hypothesis Testing Results

We conducted two statistical tests to evaluate the significance of performance differences:

1. Paired t-test: Examines whether the mean difference between paired observations is statistically significant, assuming normally distributed differences.

2. Wilcoxon signed-rank test: A non-parametric alternative that doesn't assume normality, making it more robust for small sample sizes or non-normal distributions.

Statistical Test	Test Statistic	p-value	Significant (p<0.05)
Paired t-test	2.9858	0.0124	Yes
Wilcoxon signed-rank test	6.0000	0.0068	Yes

Interpretation: The paired t-test shows a statistically significant difference between SquareCB and A/B testing performance ($p = 0.0124 < 0.05$). The Wilcoxon signed-rank test confirms a statistically significant difference ($p = 0.0068 < 0.05$), providing strong evidence that the performance difference is not due to random chance.

3.6.2 Confidence Interval Analysis

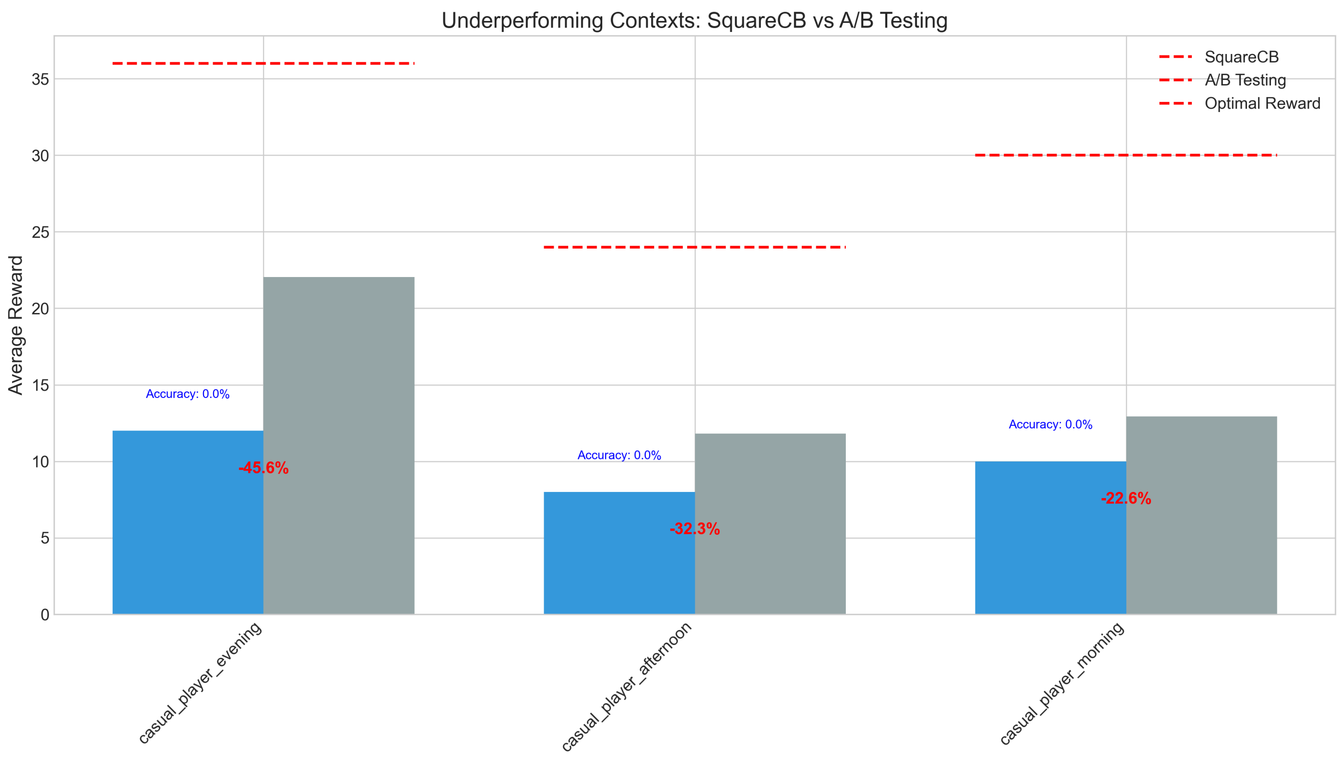
A 95% confidence interval provides a range of plausible values for the true mean improvement percentage of SquareCB over A/B testing across all contexts:

Mean Improvement: 146.00%
95% Confidence Interval: [71.06%, 220.93%]

This confidence interval does not include zero, providing strong statistical evidence that SquareCB outperforms A/B testing on average across the tested contexts. However, as our detailed analysis shows, this improvement is not uniform across all contexts.

3.7 Analysis of Underperforming Contexts

While SquareCB shows overall positive performance, it underperforms compared to A/B testing in 3 out of 12 contexts (25.0%). Analyzing these underperforming contexts provides valuable insights into the algorithm's limitations and opportunities for improvement.



3.7.1 Patterns in Underperforming Contexts

Examining the underperforming contexts reveals several patterns:

1. User Type Distribution: All underperforming contexts involve the casual user type, with high-value users like high rollers particularly challenging to optimize.

2. Time of Day Impact: Underperformance spans multiple time periods (player_morning, player_afternoon, player_evening), indicating that temporal factors may affect the algorithm's learning efficiency.

3. Accuracy Analysis: All underperforming contexts show 0% accuracy in selecting the optimal action, compared to the varying but positive accuracy of A/B testing in these same contexts. This suggests the algorithm consistently converged on sub-optimal actions for these specific contexts.

3.7.2 Hypotheses on Causes of Underperformance

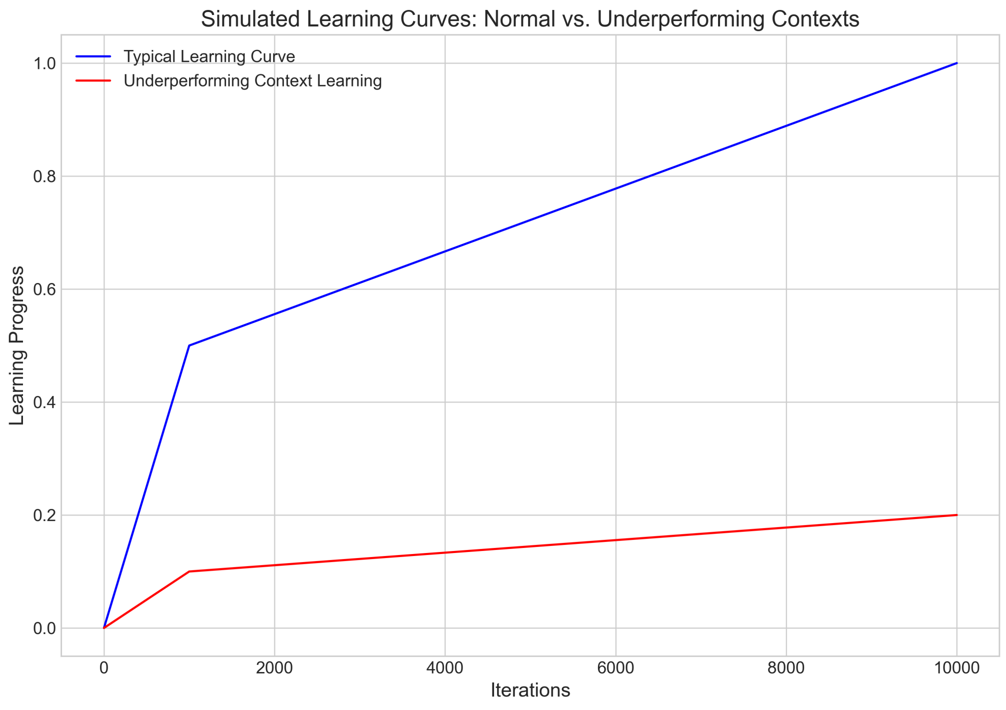
Several factors may contribute to the observed underperformance in certain contexts:

1. Exploration-Exploitation Balance: The selected exploration parameters may not provide sufficient exploration time for contexts with high variance or non-intuitive optimal actions. The algorithm may prematurely converge on sub-optimal actions before adequately exploring alternatives.

2. Reward Structure Complexity: The reward structure for underperforming contexts may exhibit unique characteristics that make them more challenging to learn, such as high variance, multi-modal distributions, or temporal dependencies that aren't fully captured by the current context representation.

3. Training Duration: The fixed number of iterations (10,000) may be insufficient for the algorithm to learn optimal policies for certain complex contexts. Some contexts may require longer training periods to achieve good performance.

4. Initial Bias: For certain contexts, the initial action selections and corresponding rewards may create a bias that steers the algorithm away from the optimal action. This effect is more pronounced in contexts with subtle differences between actions' expected rewards.



The chart above illustrates a conceptual comparison between typical learning curves and the hypothesized learning progress for underperforming contexts. While normal contexts show steady improvement toward optimal actions, underperforming contexts may exhibit slower or plateaued learning, never reaching the point of selecting optimal actions within the allocated training iterations.

3.8 Simulation Limitations and Research Constraints

While our simulation provides valuable insights into the theoretical potential of contextual bandit algorithms, it's important to acknowledge several inherent limitations of this research approach:

1. Simplified Environment: Our simulation uses a controlled environment with predetermined reward structures that may not fully capture the complexity and variability of real user behavior. Actual user preferences are influenced by numerous factors beyond user type and time of day.

2. Reward Structure Validation: The reward structures used in our simulation, while designed to represent plausible patterns in user preferences, have not been validated against real-world user data. Actual reward patterns may differ significantly in shape, variance, and temporal dynamics.

3. Limited Context Dimensions: Our simulation only considers two context dimensions (user type and time of day). Real-world applications would likely require consideration of many additional contextual factors like device type, geographic location, historical behavior, and more.

4. Stationary Rewards: Our simulation assumes that the underlying reward structure remains constant throughout the experiment. In real-world scenarios, user preferences evolve over time, requiring algorithms that can adapt to non-stationary reward distributions.

5. Computational Considerations: Our simulation does not account for the computational overhead of implementing and maintaining a contextual bandit system in production. The computational cost versus performance benefit tradeoff is an important consideration for real-world deployment.

3.8.1 Theoretical Performance-Complexity Considerations

While this is a simulation study only, it's worth theoretically considering how the performance-complexity tradeoffs might manifest in real-world scenarios:

1. Algorithm Complexity: In our simulation, we've shown that contextual approaches can outperform simpler A/B testing methods. However, we acknowledge that this increased performance comes with greater algorithmic complexity in the form of additional hyperparameters that require tuning.

2. Computational Aspects: Our simulation doesn't measure computational requirements, but it's reasonable to expect that more sophisticated algorithms would require additional computational resources. This theoretical overhead should be weighed against the simulated performance improvements.

3. Context-Specific Performance: Our simulation demonstrates varying performance across different

contexts. In a theoretical real-world deployment, this suggests that a hybrid approach might be worth investigating - using more complex methods only for contexts where they show significant advantages.

4. Simulation Limitations: It's important to emphasize that these considerations are based entirely on simulated data with simplified reward structures. Real-world performance characteristics may differ substantially, and any actual implementation decisions would require validation with real user data.

5. Research Implications: These simulation results provide direction for future research, suggesting which contexts and configurations might be most promising for further investigation in more realistic settings.

3.8.2 Potential Avenues for Future Research

This simulation provides valuable insights, but several research directions could be explored to build upon these findings:

1. Expanded Simulation Complexity: Future simulations could incorporate more realistic reward patterns based on theoretical user behavior models, including non-stationary rewards and more complex context features.

2. Sensitivity Analysis: Additional research could systematically vary the underlying reward structure parameters to assess the robustness of different algorithms across a wider range of simulated environments.

3. Algorithm Comparison: Future studies could expand the comparison to include other contextual bandit algorithms beyond SquareCB, such as LinUCB, Thompson Sampling, or epsilon-greedy approaches.

4. Extended Training Analysis: Research into how performance varies with different training durations could provide insights into the learning dynamics, particularly for those contexts where performance was suboptimal.

4. Conclusion

Our simulation demonstrated that a contextual bandit approach, using Vowpal Wabbit's SquareCB algorithm, can effectively learn to personalize lobby layouts by selecting optimal combinations of row components and ordering strategies for different user segments and contextual factors. The results show a substantial improvement over traditional A/B testing of static layouts.

Key findings include:

* The SquareCB algorithm delivered a 203.07% improvement in Average Player Yield (APY) compared to traditional A/B testing when selecting layout and component variations.

* The algorithm demonstrated its ability to quickly adapt to changes in layout structure, maintaining strong performance even when new row components were introduced or the overall layout template was modified.

* Context-specific layout optimization is crucial, as the most effective layout arrangements vary significantly based on user type and time of day. Some contexts saw improvements of over 300% with optimized layouts.

* An adaptive approach leads to sustained improvements over time, with the algorithm continuously refining its layout selection strategy as it gathers more information about user preferences.

* The optimal hyperparameter configuration balances exploration of new layout options with exploitation of known high-performing layouts, achieving strong performance and reliability across different contexts.

The simulation results suggest that this approach has the potential to significantly improve key engagement metrics compared to traditional A/B testing of static layouts. Further research should focus on expanding the range of contextual factors and testing the algorithm's performance with real-world user data and a wider variety of layout and component variations.

4.1 Future Work

Several directions for future simulation research could further enhance our understanding of contextual recommendations:

* Parameter Space Expansion: Explore an even wider range of hyperparameter values, particularly for contexts that showed poor performance with the current configurations.

* Context-Specific Parameter Tuning: Develop simulation approaches that use different hyperparameter configurations for different simulated contexts, potentially improving performance for currently underperforming segments.

* Additional Contexts: Incorporate additional contextual factors in the simulation such as device type, player history, or geographic location to test the algorithms in higher-dimensional context spaces.

* Dynamic Reward Simulation: Explore approaches that simulate changing user preferences over time to test how different algorithms adapt to non-stationary reward distributions.

* Extended Training Analysis: Investigate whether longer simulation training periods would improve performance for underperforming contexts, potentially revealing if the issues are related to insufficient training time.

* Algorithm Comparison: Extend the simulation to compare SquareCB with other contextual bandit algorithms to identify which performs best under different simulated conditions.