Technical Note: Math Adventures Adaptive Learning System

Author: Siddharth Jha

Date: November 2025

Project: AI-Powered Adaptive Math Learning Prototype

GitHub Repository: github.com/Siddharth3710/math-adaptive-prototype

1 System Overview

The **Math Adventures System** is an intelligent, rule-based adaptive learning platform designed for children ages 5–10 to practice basic mathematics. The system dynamically adjusts problem difficulty based on real-time performance using multi-factor evaluation, confidence scoring, and streak detection.

1.1 Design Philosophy

Why Rule-Based Over Machine Learning?

We chose a rule-based approach for these strategic reasons:

1. **Immediate Deployment:** Works from first use without training data.
2. **Transparency:** Every decision is explainable and auditable.
3. **Reliability:** Predictable, consistent behavior across users.
4. **Resource Efficiency:** Runs on low-end devices with no GPU.
5. **Maintainability:** Easy to tune using pedagogical feedback.

Trade-offs: Less personalized, no hidden pattern discovery, manual threshold tuning.
Future Work: Add ML layers for fine-tuning while preserving interpretability.

2 System Architecture

2.1 Component Diagram

User Interface Layer (main.py) Input Validation • Progress Display • Feedback Delivery		
Content Generator (puzzle_gen.py)	**Analytics Layer** (tracker.py)	**Decision Engine** (adaptive_engine.py)
• Problem creation • Difficulty tiers • Operation types	• Metrics calculation • Learning velocity tracking • Session persistence	• Confidence scoring • Streak detection • Multi-factor evaluation
All layers interact bidirectionally through shared session data and performance metrics.		

2.2 Data Flow

Step 1	**Step 2**	**Step 3**	**Step 4**
User Input	Validation & Question Generation	Timer Starts	Answer Submitted
⇓			
Scoring & Feedback Generation			
⇓			
Performance Recorded	**Metric Calculation**	**Confidence Estimation**	**Streak Detection**
⇓			
Decision Engine evaluates: Should difficulty adjust?			
If Yes → Adjust Difficulty		If No → Continue Same Level	
⇓			
Generate Next Question → Repeat Cycle			

3 Adaptive Logic Detailed

3.1 Core Algorithm

The adaptive engine uses a weighted scoring system:

$$\text{Adjustment Score} = \sum(Factor_i \times Weight_i \times Confidence)$$

Factors:

- **Accuracy (50% weight):** Excellent ($\geq$90%) = +3, High ($\geq$80%) = +2, Low ($\leq$50%) = -2
- **Streak Bonus (30% weight):** Hot streak (3+) = +min(2, streak/3); Cold streak = -min(2, streak/3)
- **Speed Modifier (20% weight):** Fast (<5s) + accurate = +1; Slow (>15s) = -0.5

Final Score adjusted by Confidence multiplier (0.5x–1.2x).

3.2 Confidence Scoring

$$\text{Confidence} = 0.5(\text{Accuracy}) + 0.25(1 - \frac{\text{Var}}{100}) + 0.25(1 - \frac{\text{Alternations}}{\text{Attempts}})$$

Interpretation:

- High (>0.8): Aggressive adjustments
- Medium (0.4–0.8): Normal adjustments
- Low (<0.4): Conservative, larger evaluation window

3.3 Dynamic Window Sizing

$$\text{Window Size} = 3 + \text{Difficulty Factor} + \text{Confidence Factor}$$

Range: 2–5 questions.

Examples:

- Easy + High Confidence = 2
- Hard + Low Confidence = 5

3.4 Streak Detection

Hot Streak (3+ correct): Increases difficulty and confidence. **Cold Streak (3+ wrong):** Decreases difficulty, encourages user.

4 Metrics Tracked

4.1 Real-Time Metrics

Metric	Purpose	Frequency
Accuracy	Primary difficulty indicator	Per attempt
Response Time	Speed/confidence measure	Per attempt

Streak Length	Detect mastery/struggle	Per attempt
Confidence	Decision certainty	Every 2–3 attempts
Window Size	Adaptation timing	Per evaluation

4.2 Session-Level Metrics

Metric	Calculation	Usage
Learning Velocity	$(2^{nd}_{half} - 1^{st}_{half})$	Trend detection
Operation Breakdown	Accuracy/time per operation	Identify weaknesses
Difficulty Progression	Track transitions	Visualization
Consistency Score	Variance of accuracy/time	Stability check

4.3 Example Decision

```
Recent: 3 correct (100%), avg time 4.4s
Streak: 3, Confidence = 0.89
Adjustment Score = (3 + 2 + 1) × 1.2 = 7.2
Decision: Increase difficulty → HARD
Reason: Outstanding! High confidence (89%)
```

5 Performance Tracking Innovations

5.1 Operation Analytics

```
Addition:       [] 80% (4/5)
Subtraction:    [] 100% (3/3)
Multiplication: [] 95% (4/4)
Division:       [] 67% (2/3)
```

5.2 Learning Velocity

$$V = 0.7(\text{Accuracy Change}) + 0.3(\text{Speed Improvement})$$

Interpretation: $V > 0.1$: Improving $-0.1 \leq V \leq 0.1$: Stable $V < -0.1$: Declining

5.3 Session Persistence

Sessions saved as JSON for tracking, progress analysis, and ML training.

6 User Experience Enhancements

- **Visual Progress:** ASCII charts for difficulty trend.
- **Contextual Feedback:** Encouraging messages based on streaks and accuracy.
- **Milestones:** Every 5 questions, show progress summary.

7 Edge Case Handling

Noisy Performance: Reduces confidence, increases evaluation window.
Insufficient Data: Uses default confidence (0.5) and conservative bias.
Boundary Conditions: Prevents difficulty beyond Easy/Hard.

8 Validation and Testing

- Puzzle Generation: All operations and levels
- Adaptive Logic: Threshold and score calculation
- Confidence Scoring: Handles variance cases
- Integration: End-to-end simulations

9 Future Work

Short-Term: Spaced repetition, dashboard, mobile app. **Long-Term:** Hybrid ML approach, Bayesian Knowledge Tracing, NLP-based problems.

10 Conclusion

This adaptive system demonstrates effective personalization through transparent, rule-based logic without heavy ML. It ensures:

- Interpretability for educators
- Reliability for learners
- Flexibility for ML enhancement
- Portability across devices

Appendix A: Threshold Justification

Threshold	Value	Rationale
Accuracy High	0.8	Standard mastery threshold in education
Accuracy Low	0.5	Indicates struggle
Time Fast	5s	Elementary mental math target
Streak Limit	3	Statistically significant repetition
Confidence High	0.8	Certainty threshold for adjustment

Document Version: 1.0
Total Pages: 2–3 (formatted)