Contents

Monte Carlo Simulation Lesson - Teacher Guide

Overview

This lesson introduces students to Monte Carlo simulations through calculating π using random sampling. It's designed for 11th-12th grade students with strong math backgrounds and provides an accessible entry point to computational thinking and probabilistic methods used in AI/ML.

Duration: 2-3 class periods (90-135 minutes total) **Prerequisites:** - Basic Python programming (variables, functions, loops) - Coordinate geometry (Cartesian plane, distance formula) - Basic probability concepts - Familiarity with Jupyter notebooks

---

Learning Objectives

By the end of this lesson, students will be able to:

1. **Explain** what Monte Carlo simulations are and why they're useful
2. **Implement** a Monte Carlo algorithm to estimate π
3. **Analyze** how sample size affects accuracy (convergence)
4. **Evaluate** the statistical properties of probabilistic algorithms
5. **Connect** Monte Carlo methods to real-world applications in AI and data science

---

Lesson Structure

Day 1: Introduction and Core Concepts (45 minutes)

Opening (10 minutes) 1. Hook: Show the dartboard visual or physical demonstration 2. Ask: "How could we calculate π if we didn't know the formula?" 3. Introduce Monte Carlo concept through the casino analogy

Direct Instruction (15 minutes) 1. Explain the geometry: circle inscribed in square 2. Walk through the mathematical relationship 3. Demonstrate why ratio of points gives us π 4. Show convergence concept with small demo

Guided Practice (20 minutes) Students work through Parts 1-4 of the notebook: - Task 1: Generate random points - Task 2: Check if points are inside circle - Task 3: Calculate π estimate

Circulate and Support: - Check for understanding of coordinate ranges (-1 to 1) - Ensure students understand the distance formula - Help debug common errors (see "Common Issues" section)

Day 2: Analysis and Extensions (45-60 minutes)

Review (5 minutes) - Quick recap of previous day - Show visualization from Part 6 - Discuss initial observations about accuracy

Independent Work (25 minutes) Students complete: - Task 4: Convergence analysis - Task 5: Multiple simulations

Class Discussion (15 minutes) Guide discussion using these questions: 1. What pattern do you notice as n increases? 2. Why doesn't the estimate ever become *exactly* π? 3. How many samples would you need for your application? 4. What trade-offs exist between accuracy and computation time?

Extension Activities (15 minutes if time) - Challenge: 3D sphere version - Research: Real-world Monte Carlo applications - Connect to AI: How do neural networks use randomness?

Day 3 (Optional): Applications and Presentations (45 minutes)

Student Presentations (30 minutes) - Small groups share their convergence findings - Discuss different approaches to the 3D challenge - Share research on real-world applications

Synthesis (15 minutes) - Connect to broader AI/ML concepts - Discuss ethical considerations of probabilistic algorithms - Preview next topics in data science curriculum

---

Differentiation Strategies

For Students Who Need Support

1. **Pre-teaching Session:**
   - Review coordinate geometry
   - Practice with `np.random.uniform()`
   - Clarify function syntax and return values
2. **Scaffolded Notebook Version:**
   - Provide more code hints in TODO comments
   - Include pseudocode for each function
   - Break Task 3 into smaller sub-tasks
3. **Visual Aids:**
   - Physical dartboard demonstration
   - Large printout of circle/square diagram
   - Step-by-step flowchart of algorithm
4. **Peer Support:**
   - Pair with stronger programmer
   - Provide worked example for similar problem
   - Extra office hours availability

For Advanced Students

1. **Extension Challenges:**
   - Implement 3D version (already in notebook)
   - Calculate confidence intervals
   - Compare different random number generators
   - Optimize code for speed using NumPy vectorization
2. **Research Projects:**
   - Investigate MCMC (Markov Chain Monte Carlo)
   - Apply to real dataset (e.g., estimating integral)
   - Create visualization comparing multiple methods
   - Write blog post explaining Monte Carlo to general audience
3. **AI Connections:**
   - Read about stochastic gradient descent
   - Explore Monte Carlo tree search in game AI
   - Investigate dropout in neural networks
   - Study uncertainty quantification in ML models

Assessment

Formative Assessment

During Class: - Observe debugging strategies - Check understanding through targeted questions - Review intermediate outputs (do arrays look reasonable?) - Monitor collaboration and problem-solving approaches

Check for Understanding: - Can students explain why we multiply by 4? - Do they recognize that more samples → better accuracy? - Can they interpret error messages and fix bugs?

Summative Assessment (100 points)

Automated Grading (80 points) Use the provided `grade_monte_carlo.py` script:

```
python grade_monte_carlo.py student_notebook.ipynb
```

Breakdown: - Task 1: `generate_random_points()` - 20 points - Task 2: `is_inside_circle()` - 20 points
- Task 3: `estimate_pi()` - 25 points - Task 4: `analyze_convergence()` - 15 points - Task 5: `run_multiple_simulations()` - 15 points - Code style and documentation - 5 points

Reflection Questions (20 points) Part 9 of notebook: - Question 1: Understanding convergence (5 points) - Question 2: Interpreting distributions (5 points) - Question 3: Quantitative analysis (5 points) - Question 4: Real-world connections (5 points)

Rubric for Reflection Questions: - **Exceptional (5):** Sophisticated understanding, connects to broader concepts - **Proficient (4):** Clear understanding, accurate explanation - **Developing (3):** Partial understanding, some misconceptions - **Emerging (2):** Limited understanding, significant gaps - **Incomplete (0-1):** Little to no understanding, or not attempted

---

Common Issues and Solutions

Technical Issues

Issue 1: "ModuleNotFoundError: No module named 'numpy'" - **Solution:** Install required packages: `pip install numpy matplotlib` - **Prevention:** Provide environment setup instructions before class

Issue 2: Plots not displaying - **Solution:** Add `%matplotlib inline` to first code cell - **Alternative:** Use `plt.show()` explicitly

Issue 3: Notebook kernel crashes - **Cause:** Usually from running too many points (>10 million) - **Solution:** Restart kernel, use smaller sample sizes - **Prevention:** Warn students about computational limits

Conceptual Issues

Issue 1: "Why do we multiply by 4?" - **Solution:** Draw the area ratios diagram again - **Hint:** Circle area = $\pi r^2 = \pi(1)^2 = \pi$; Square area = 4 - **Key insight:** Ratio of areas = $\pi/4$, so $\pi = 4 \times$ ratio

Issue 2: "My estimate is way off from π" - **Check:** Are points being generated in correct range? - **Check:** Is the distance formula correct ($x^2 + y^2 \leq 1$)? - **Check:** Is the calculation 4 × (inside/total) correct? - **Reassure:** Small sample sizes will have high error

Issue 3: "Why doesn't it give exactly π?" - **Explanation:** This is a probabilistic method - **Analogy:** Like estimating average height by measuring 100 people - **Connect to:** Law of Large Numbers (approaches true value as n → ∞)

Coding Issues

Issue 1: Function returns wrong number of values

```
# Wrong:
return x, y, inside  # Only 3 values

# Correct:
return pi_estimate, x, y, inside  # 4 values
```

Issue 2: Forgetting to convert boolean to count

```
# Wrong:
pi_estimate = 4 * inside / n  # inside is array

# Correct:
pi_estimate = 4 * np.sum(inside) / n  # Count True values
```

Issue 3: Off-by-one errors in ranges

```
# Wrong:
x = np.random.uniform(0, 1, n)  # Only positive values

# Correct:
x = np.random.uniform(-1, 1, n)  # Full range
```

Connection to AI Fellows Program Goals

This lesson directly supports the AI Fellows mission in several ways:

1. Transdisciplinary Integration

- **Math:** Geometry, probability, statistics
- **Computer Science:** Algorithms, complexity analysis
- **Physics:** Modeling, simulation methods
- **Data Science:** Sampling, convergence, visualization

2. Practical AI/ML Concepts

Students learn foundational concepts used in: - Neural network training (stochastic methods) - Reinforcement learning (Monte Carlo value estimation) - Bayesian inference (MCMC sampling) - Uncertainty quantification

3. Accessible Entry Point

- No advanced AI prerequisites needed
- Visual and intuitive approach
- Immediate feedback through visualizations
- Clear connection to familiar concept (π)

4. Scalable Implementation

- Works in any Python environment
- Minimal dependencies (numpy, matplotlib)
- Self-contained lesson
- Easy to adapt for different grade levels

5. Real-World Relevance

Students can research and present on: - Financial risk modeling - Climate change prediction - Drug development simulations - Game AI (Monte Carlo tree search)

Additional Resources

For Teachers

Background Reading: - "Monte Carlo Methods in Practice" (Wikipedia) - "The History of Monte Carlo Methods" - Nick Metropolis - NCTM article: "Monte Carlo Methods in the Classroom"

Video Resources: - 3Blue1Brown: "Simulating an Epidemic" - Numberphile: "Monte Carlo Simulations" - Khan Academy: "Law of Large Numbers"

Extension Problems: - Estimating integrals using Monte Carlo - Buffon's Needle Problem - Monte Carlo optimization examples

For Students

Beginner-Friendly: - How Random Numbers Help Us Understand the World - The Monte Carlo Method explained to kids - Why casinos always win (probability concepts)

Advanced: - Introduction to Computational Thinking (MIT OpenCourseWare) - Monte Carlo Methods in Machine Learning - Scientific Computing with Python

Interactive Demos

Recommended Websites: - PhET Interactive Simulations (University of Colorado) - Seeing Theory (Brown University) - probability visualizations - Desmos calculator activities

---

Adapting for Different Contexts

For Virtual/Hybrid Learning

Synchronous Session: - Use breakout rooms for pair programming - Share screen for live coding demonstrations - Poll students on key concepts - Use collaborative notebooks (Google Colab)

Asynchronous Options: - Record video walkthrough of each task - Provide discussion board for questions - Create Loom videos showing debugging techniques - Office hours via Zoom for support

For Lower Grade Levels (9th-10th)

Simplifications: - Pre-fill more of the code (provide skeleton functions) - Focus on Parts 1-6 only (skip statistical analysis) - Use smaller sample sizes to reduce wait time - More guided practice before independent work

Modified Objectives: - Understand basic concept of random sampling - Implement simple Python functions - Interpret visualizations - Appreciate that more data → better estimates

For Integration with Other Subjects

Physics Class: - Connect to kinetic theory of gases - Discuss particle simulations - Relate to measurement uncertainty - Link to experimental design

Statistics Class: - Emphasize Law of Large Numbers - Explore sampling distributions - Calculate confidence intervals - Compare to other estimation methods

Computer Science Class: - Analyze algorithm complexity - Compare different random number generators - Optimize for speed using profiling - Explore parallel computing approaches

---

Sample Timeline for AI Fellows Curriculum

This lesson fits naturally into a sequence:

Week 1-2: Introduction to Python and Jupyter **Week 3-4:** Data visualization basics **Week 5-6:** **Monte Carlo simulations** ← This lesson **Week 7-8:** Introduction to machine learning concepts **Week 9-10:** Simple neural network from scratch **Week 11-12:** Real-world AI applications project

---

Grading Tips

Using the Automated Grader

1. **Setup:**

```
# Ensure dependencies are installed
pip install numpy matplotlib

# Run the grader
python grade_monte_carlo.py student_notebook.ipynb
```

2. **Review Report:**
    - Check `student_notebook_grade_report.txt`
    - Review any failed test cases
    - Provide targeted feedback for corrections
3. **Manual Review:**
    - Always check reflection questions manually
    - Review code style beyond what script checks
    - Consider partial credit for effort
    - Look for creative approaches

Grading Reflection Questions

Look for: - Evidence of testing and observation - Connection to mathematical concepts - Proper use of vocabulary (convergence, distribution, etc.) - Thoughtful real-world applications

Common Excellent Responses: - Question 1: Discusses Law of Large Numbers, diminishing returns - Question 2: Notes normal-ish distribution, centered on π - Question 3: Provides specific number with reasoning - Question 4: Gives detailed, relevant real-world example

Grade Adjustments

Consider bonuses for: - Completing 3D extension challenge - Adding creative visualizations - Optimizing code beyond requirements - Helping peers debug effectively

Reasons for partial credit: - Functions work but are inefficient - Small logical errors that don't break code - Missing edge case handling - Documentation could be clearer

---

Facilitating Productive Struggle

When to Step In

Intervene when: - Student has been stuck >10 minutes on same error - Frustration is leading to disengagement - Fundamental misconception is blocking progress - Syntax error they can't identify

Hold back when: - Student is actively debugging and making progress - Error messages are clear and student hasn't read them - Pair is collaborating effectively - Productive struggle is happening

Questioning Strategies

Instead of: "That's wrong, here's how to fix it" **Try:** "What do you expect this line to do? What's it actually doing?"

Instead of: "You need to use np.sum()" **Try:** "What type is `inside`? What do you need it to be?"

Instead of: "The range should be -1 to 1" **Try:** "Can you show me on the diagram where your points are landing?"

Debugging Together

Model the process: 1. "Let's print out what we're getting..." 2. "Does this match what we expect?" 3. "Where might the problem be?" 4. "How can we test this specific part?"

Assessment Data Analysis

After grading all submissions, analyze patterns:

Class Performance Metrics

Calculate: - Average score on each task - Most common errors - Time spent on different sections - Correlation between tasks (do students strong in Task 1 do well in Task 5?)

Use insights to: - Adjust future pacing - Add more practice on weak areas - Refine explanations - Identify students needing additional support

Individual Feedback

Provide specific comments on: - What they did well - One key area for improvement - Connection to broader learning goals - Next steps for growth

Example feedback: > "Excellent work on the core algorithm! Your convergence analysis was thorough. For future projects, try adding more descriptive variable names and comments to make your code even more readable. Consider exploring how these concepts apply to machine learning optimization."

Safety and Ethics Discussion

Optional Extension: Ethical Considerations

Discussion prompts: 1. "Monte Carlo methods are used in finance to price derivatives. What are ethical implications of using probabilistic models for important decisions?"

2. "AI systems often use randomness in training. Should we be concerned that results aren't completely reproducible?"
3. "Self-driving cars use Monte Carlo simulations to predict outcomes. What level of accuracy is 'good enough' for safety-critical applications?"

Learning objectives: - Understand limitations of probabilistic methods - Consider when approximations are vs. aren't acceptable - Think critically about AI in decision-making - Appreciate importance of uncertainty quantification

---

Success Metrics

For Students

✓ Can explain Monte Carlo concept to a peer ✓ Successfully implement all required functions ✓ Understand relationship between sample size and accuracy ✓ Connect to real-world applications ✓ Feel confident with basic Python data science workflow

For Teachers

✓ 80%+ of students complete core tasks (Parts 1-6) ✓ Class discussion shows deep understanding ✓ Students can explain *why* method works, not just *how* ✓ Positive engagement and curiosity about applications ✓ Skills transfer to subsequent data science lessons

---

Contact and Support

For questions about this lesson or the AI Fellows program: - Check Canvas module for discussion forums - Attend monthly AI Fellows collaboration sessions - Email program coordinator for technical support - Share adaptations and improvements with fellow teachers

---

Remember: The goal is not just to calculate π, but to introduce students to computational thinking and probabilistic methods that underpin modern AI and data science. Focus on understanding and application, not just getting the "right answer."

Good luck, and happy teaching!