One-Week Markov Chains Syllabus (Go Language Focus)

Day 1 — Introduction & The Markov Property

Learning Goals:
- Understand what a Markov chain is.
- Define states, transitions, and probabilities.
- Internalize the Markov property (memoryless).
- See the connection between probability distributions and state transitions.

Math:
- Definition: A stochastic process {X_t} is a Markov chain if
P(X_{t+1} = j | X_t = i, X_{t-1}, ..., X_0) = P(X_{t+1} = j | X_t = i).
- Transition matrix P, where P_{ij} = P(X_{t+1}=j | X_t=i).
- Visualize state diagrams (nodes and directed edges).

Go Programming:
- Represent states using constants or enums.
- Use a map[string]map[string]float64 for transition probabilities.
- Write a function that simulates the next state given the current state.

Day 2 — Transition Matrices & Evolution of Distributions

Learning Goals:
- Represent a Markov chain with a transition matrix.
- Predict future distributions using matrix multiplication.
- Understand P^n as n-step transition probabilities.

Math:
- If distribution at time t is p_t, then: p_{t+1} = p_t P
- n-step distribution: p_n = p_0 P^n

Go Programming:
- Use slices of slices ([][]float64) to represent matrices.
- Write functions for vector-matrix multiplication.
- Experiment with repeated multiplication to simulate n-step transitions.

Exercises:
- Given a 3-state chain, compute the probability of being in state 2 after 5 steps.

Day 3 — Stationary Distributions & Long-Run Behavior

Learning Goals:
- Understand what stationary distributions are.
- Learn conditions for convergence.
- Connect simulation results with analytical stationary distributions.

Math:
- Stationary distribution π satisfies: $\pi P = \pi$, sum_i $\pi_i = 1$
- Concepts: irreducible, aperiodic, ergodic.

Go Programming:
- Simulate long sequences and track state frequencies.
- Compare empirical frequencies to approximate π.
- (Optional) Use Gonum to solve for π using linear algebra.

Exercises:

- Implement a 3-state chain and compute both simulation-based and algebraic stationary distributions.

Day 4 — Absorbing Chains & Expected Times

Learning Goals:
- Learn about absorbing states and transient states.
- Calculate expected time to absorption.
- Apply Markov chains to board games, random walks, etc.
Math:
- Absorbing Markov chain canonical form:
P = [[I, 0], [R, Q]]
- Fundamental matrix: N = (I - Q)^(-1)
- Expected steps until absorption: row sums of N.
Go Programming:
- Write a simulation of a gambler's ruin problem with absorbing states.
- Estimate expected absorption time empirically.
- Implement basic matrix operations to compute theoretical values.
Exercises:
- Model the probability that a gambler with 10 coins goes broke before reaching 20.

Day 5 — Markov Chain Monte Carlo (MCMC)

Learning Goals:
- Understand why Markov chains are used for sampling.
- Learn the basics of the Metropolis algorithm.
- Connect Markov chains to probability distributions.
Math:
- Target distribution $\pi(x)$.
- Proposal distribution $q(x'|x)$.
- Acceptance probability: $\alpha = \min(1, [\pi(x')q(x|x')] / [\pi(x)q(x'|x)])$.
Go Programming:
- Implement a Metropolis sampler using math/rand for randomness.
- Store generated samples in slices.
- Compare histogram of samples to target distribution.
Exercises:
- Modify the sampler to handle a bimodal distribution.

Day 6 — Advanced Theory: Convergence & Mixing

Learning Goals:
- Understand eigenvalues & convergence speed.
- Learn about mixing time and how chains "forget" the start state.
- Explore reducibility and periodicity.
Math:
- Second-largest eigenvalue modulus (SLEM) controls convergence rate.
- Reducible chains: not all states communicate.
- Periodic chains: chain cycles instead of converging.
Go Programming:
- Write functions to check reducibility (graph connectivity).
- Write functions to check periodicity.

- Plot convergence (e.g., using gonum/plot or exporting data for external plotting).

Day 7 — Mini-Project

Goal: Tie math + code together with an applied project.
Options:
1. Text Generation (Markov chain on text).
- Parse text and build transition probabilities between words.
- Generate random sentences.
2. Random Walk in 2D
- Simulate paths with goroutines for parallel experiments.
- Visualize with ASCII output or Gonum plotting.
3. PageRank Algorithm
- Implement PageRank using transition matrices in Go.
Deliverable:
- A well-structured Go program with:
- Clear functions for transitions, simulation, and analysis.
- Documentation of results.
- (Optional) Visualization with gonum/plot.

Mastery Resources (for after the week)

- Gonum (https://gonum.org) for matrix computations in Go.
- "Markov Chains" by Norris (theory).
- "Introduction to Probability" by Bertsekas & Tsitsiklis (applications).