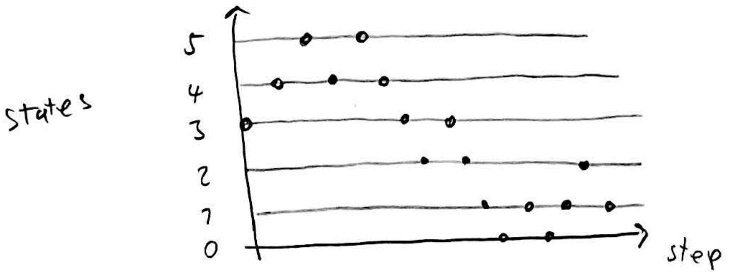
(x) End lecture Oct 29.

Markov Chain (Andrey Markov)

- A random sequence in which the prob. for the next state only depends on the current state, not the history of states before that. (Markov property, memoryless)

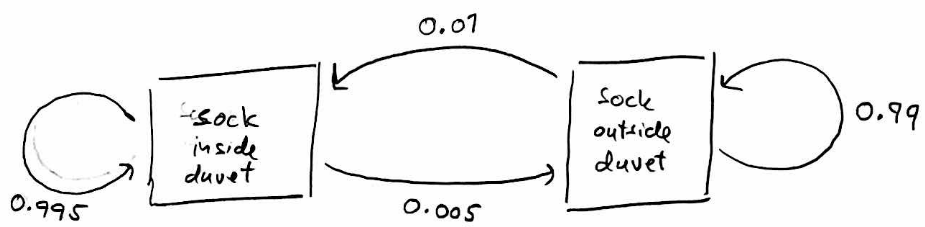
$$\left[\, p(x_{i+1} \mid x_i, x_{i-1}, \dots) = p(x_{i+1} \mid x_i) \,\right]$$

Examples: "Drunkards walk": move +1 or -1 with equal prob.

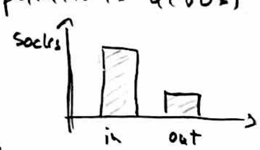


[Probs. for next number dep. only on current number, not how the chain got there.]

"Socks in duvet"



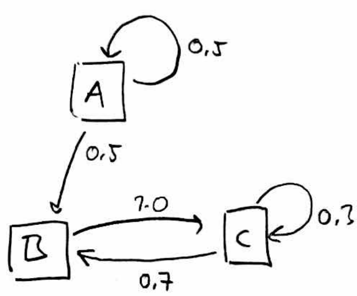
- Distribute walkers across states and iterate → evolve towards stationary/equilib. populations across states



- Important concept: Ergodicity

  A chain is ergodic if any state can be reache from any other state with a finite number of steps

Non-ergodic chain:



Markov Chain Monte Carlo

[A method that produces a Markov chain of samples / events / steps whose stationary distribution is a given prob. distribution]

In other words: Generate a set of samples $x_i \sim p(x)$ by recording the steps of a Markov chain in x-space.

General procedure:

- Current state x_i
- Generate a new candidate x' using a proposal pdf that only depends on current state
- Apply acceptance rule. If accepted: $x_{i+1} = x'$
  If rejected: $x_{i+1} = x_i$
- Repeat

For the list of samples $\{x_1, x_2, x_3, \ldots\}$ to end up as a set of samples distributed according to $p(x)$, the acceptace rule should ensure

- Ergodicity (can visit all states)
- Detailed balance: $p(x_i)\,\mathrm{Prob}(x_i \to x') = p(x')\,\mathrm{Prob}(x' \to x_i)$

can have: small large = large small

- The Metropolis (or Metropolis-Hastings) algorithm is one example that ensures this. [Basis: $\text{Prob}(x \to x') = T(x \to x') A(x \to x')$]

  - Generate candidate according to some <u>proposal prob.</u> $T(x_i \to x')$

  - Calculate <u>acceptance prob.</u>

$$A(x_i \to x') = \min\left(1, \frac{p(x')}{p(x_i)} \frac{T(x' \to x_i)}{T(x_i \to x')}\right)$$

If proposal is symmetric: $T(x' \to x_i) = T(x' \to x_i)$ (Metropolis algo.)

$$A(x_i \to x') = \min\left(1, \frac{p(x')}{p(x_i)}\right)$$

  - Generate random number r from $U(0,1)$
    - If $r \leq A(x_i \to x')$, then <u>accept</u>: $x_{i+1} = x'$
    - If $r > A(x_i \to x')$, then <u>reject</u>: $x_{i+1} = x_i$

- <u>Note:</u>
  - <u>Always</u> accepts jump to state with higher prob. ($p(x') > p(x)$)
  - <u>Sometimes</u> accepts jump to state with lower prob ($p(x') < p(x)$) (with prob. that ensures detailed balance.)

[Wait long enough $\Rightarrow$ all states should be explored (theoretically). Freedom in choosing proposal prob. dist.]

  - Only depends on <u>ratio</u> $\frac{p(x')}{p(x)}$, so normalization constants (like the partition function) <u>cancel</u>! [Very important in proj. 4!]

- Suggested algorithm for Project 4 :

Repeat N times = "one MC cycle"

due to strong correlations between subseq. samples in the state space.

- Generate candidate state $\bar{s}'$
  - Pick random spin from lattice
  - Flip it!
- Find value of $\frac{p(\bar{s}')}{p(\bar{s}_i)}$ [In an efficient way! Don't evaluate $p(\bar{s}')$ and $p(\bar{s})$ separately, and don't evaluate $exp(\dots)$ all the time!]
- Generate random number $r \sim U(0,1)$ and carry out accept/reject step : accept if $r < \frac{p(\bar{s}')}{p(\bar{s}_i)}$

- Then use the new state $\bar{s}_{i+1}$ to compute energy, magnetization, expectation values after this number of cycles, etc.
- Store relevant numbers
- Repeat

Two views on the MCMC in Project 4

① As model of (discretized) time evolution of system

- Each time step there are $\sim N$ random spin-environment interactions that can flip the spin. So the Markov chain

$$\bar{s}_0 \to \bar{s}_1 \to \bar{s}_2 \to \bar{s}_3 \to \ldots$$

"effectively" means

$$\bar{s}(t_0) \to \bar{s}(t_1) \to \bar{s}(t_2) \to \bar{s}(t_3) \to \ldots$$

- If system starts in state $\bar{s}_0$ that has low prob. under equilibrium assumption (Boltzmann), then letting it evolve will "equilibrate the system", i.e. take $\bar{s}$ into regions of state space that are more likely in equilibrium.

  ↳ Connects to MCMC "burn-in"

  ↳ Some unfortunate terminology in the literature, which makes it sound like $p(\bar{s})$ is changing, when its just the chain of $\bar{s}$ samples that is converging towards a good approx to $p(\bar{s})$.

---

② As a method for sampling from $p(\bar{s})$

- Just some way to draw a set of samples $\bar{s}_1, \bar{s}_2, \bar{s}_3, \ldots$ such that the distribution of samples (in the long run) corresponds to $p(\bar{s})$.

- Not directly related to the physics — equally applicable for any pdf, regardless of topic.

- The physics of equilibrium was already contained in our choice to use the Boltzmann dist. for $p(\bar{s})$

Burn-in

- Samples in MCMC are correlated, i.e. not "independent and identically dist. draws" (iid)

$$p(x_{i+1} | x_i) \neq p(x_{i+1})$$

- In the limit $n \to \infty$, we will have $\frac{n_{x \in [x, x+dx]}}{n} = p(x)dx$

  number of samples

- But we cannot do $n \to \infty$ in practice!

  $\Rightarrow$ Our resulting distribution of samples (and hence our estimated expectation values) depends on starting point, which can be unreasonably improbable compared to the number of samples we draw!

- Usual solution:

  - Throw away first part of a chain of samples (burn-in)
  - Not use every new sample (even after burn in)
  - Run many chains from different starting points and combine results.