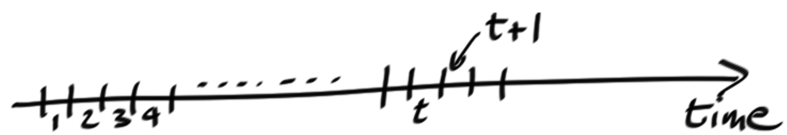
Markov processes – I

- checkout counter example
- Markov process definition
- n-step transition probabilities
- classification of states

past   future

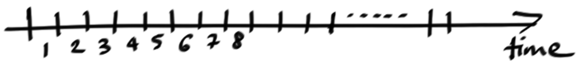
STATE



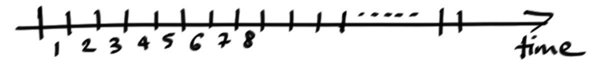
$$state(t+1) = f(state(t), noise)$$

checkout counter example

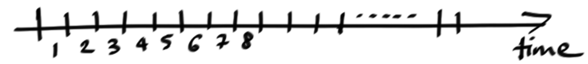
- discrete time $n = 0, 1, \ldots$



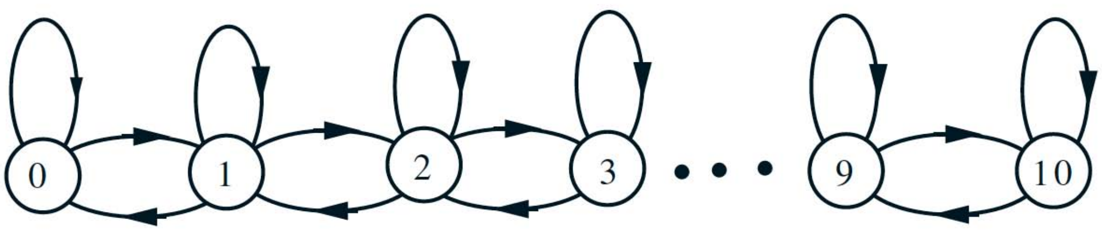
- customer arrivals: Bernouilli(p)



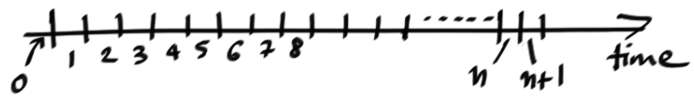
- customer service times: geometric(q)



- "state" X_n: number of customers at time n

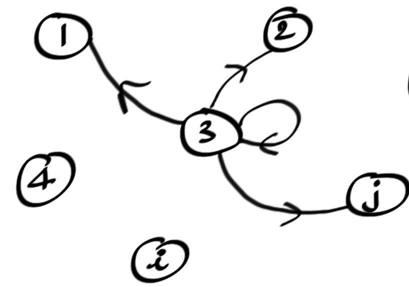


discrete-time finite state Markov chains



- X_n: state after n transitions
  - belongs to a finite set
  - initial state X_0 either given or random
  - transition probabilities:

$$\begin{aligned} p_{ij} &= \mathbf{P}(X_1 = j \mid X_0 = i) \\ &= \mathbf{P}(X_{n+1} = j \mid X_n = i) \end{aligned}$$



- Markov property/assumption:
  "given current state, the past doesn't matter"

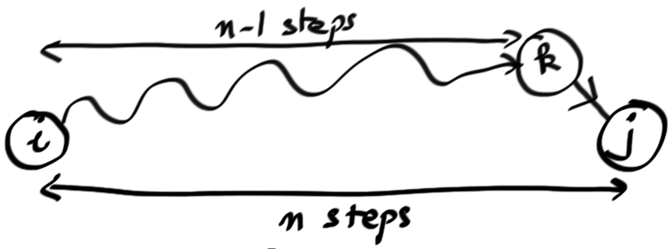
$$\begin{aligned} p_{ij} &= \mathbf{P}(X_{n+1} = j \mid X_n = i) \\ &= \mathbf{P}(X_{n+1} = j \mid X_n = i,\ X_{n-1}, \ldots, X_0) \end{aligned}$$

- model specification: identify states, transitions, and transition probabilities

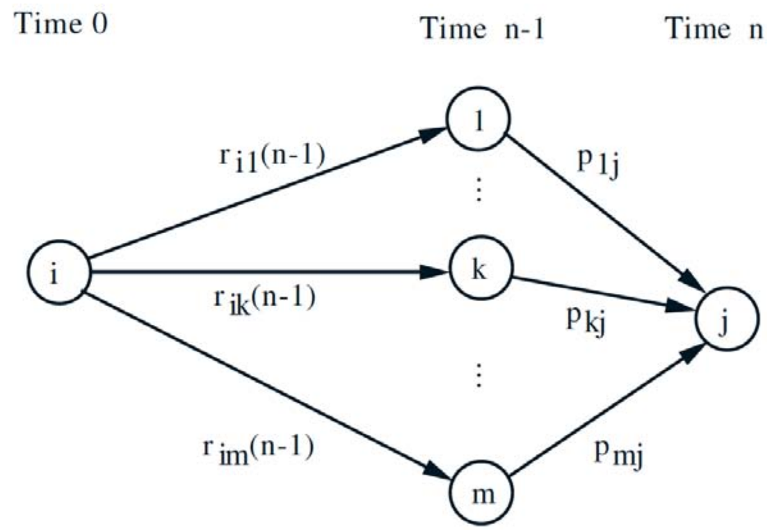
n-step transition probabilities

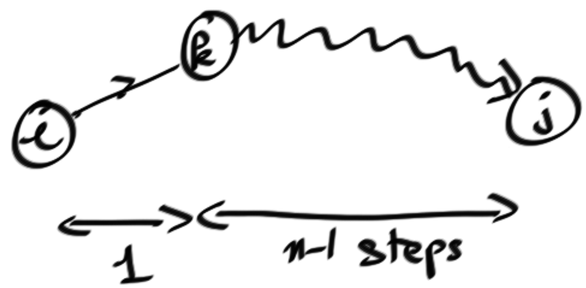
- state probabilities, given initial state i:

$$\begin{aligned} r_{ij}(n) &= \mathbf{P}(X_n = j \mid X_0 = i) \\ &= \mathbf{P}(X_{n+s} = j \mid X_s = i) \end{aligned}$$



- key recursion: $r_{ij}(n) = \sum_{k=1}^{m} r_{ik}(n-1)p_{kj}$

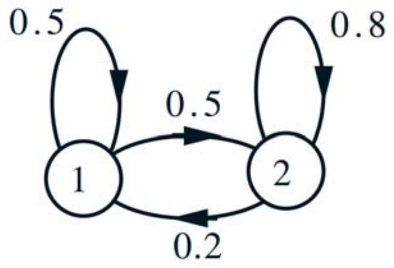




- random initial state:

$$\mathbf{P}(X_n = j) = \sum_{i=1}^{m} \mathbf{P}(X_0 = i) r_{ij}(n)$$

example

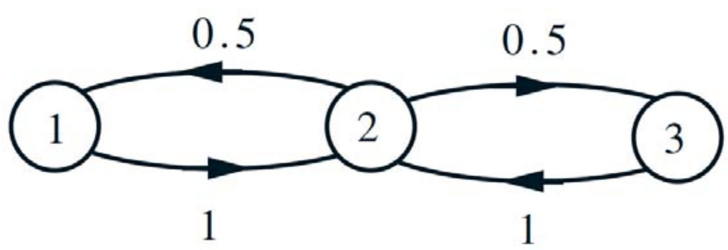


$$r_{ij}(n) = \mathbf{P}(X_n = j \mid X_0 = i)$$

	$n=0$	$n=1$	$n=2$	$n=100$	$n=101$
$r_{11}(n)$					
$r_{12}(n)$					
$r_{21}(n)$					
$r_{22}(n)$					

generic convergence questions

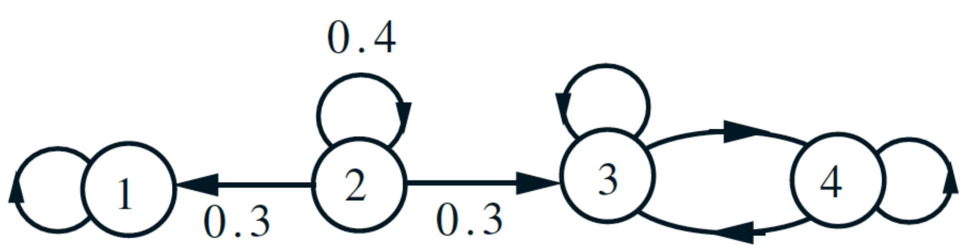
- does $r_{ij}(n)$ converge to something?



n odd: $r_{22}(n) =$

n even: $r_{22}(n) =$

- does the limit depend on initial state?



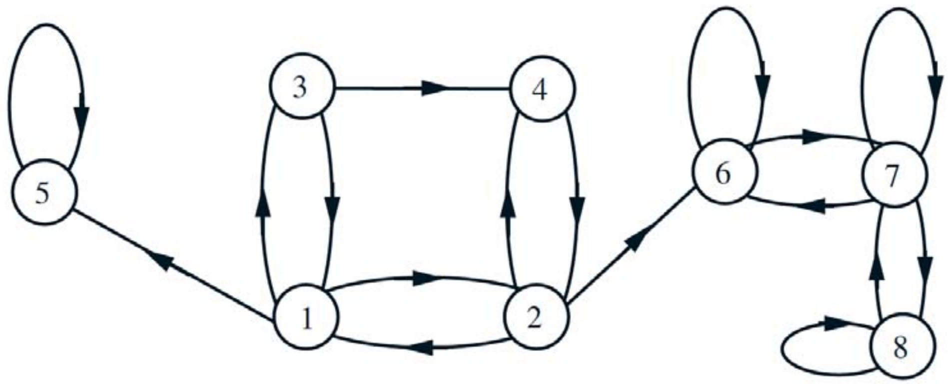
$r_{11}(n) =$

$r_{31}(n) =$

$r_{21}(n) =$

recurrent and transient states

- state i is recurrent if "starting from i, and from wherever you can go, there is a way of returning to i"
- if not recurrent, called transient



- recurrent class: a collection of recurrent states communicating only between each other