Discrete Markov Chains

April 23, 2015

1 Lecture 1

Definitions and General Properties We are discussing a stochastic process

$$x_k,\ k = 0, 1, \dots,$$

where each x_k is a random variable mapping into a finite state space

$$S = \{S_1, \dots, S_n\}.$$

We will also denote the states by $i, j, k, ...$ in what follows. The discrete time step k can be abstract or can refer to some physical time τ. Such a process is called a Markov chain if the following property holds:

Definition 1. A stochastic process as above satisfies the **Markov property** if for all $k \geq 1$ and states $S_0, \dots, S_k$:

$$\mathbb{P}(x_k = S_k | x_{k-1} = S_{k-1}, \dots, x_0 = S_0) = \mathbb{P}(x_k = S_k | x_{k-1} = S_{k-1}),$$

i.e. the distribution of the chain knowing the entire history of the process is identical to the distribution knowing only the last step. In short, we will write

$$\mathbb{P}(x_k | x_{k-1}, \dots, x_0) = \mathbb{P}(x_k | x_{k-1}).$$

Throughout the lecture, we will only be interested in **time-homogeneous** chains. In time-homogeneous Markov chains, there is no dependence on time, and the transition probabilities depend on the states exclusively. In this case we can define a transition probability matrix, or short, **transition matrix**:

$$\mathbf{T} \in \mathbb{R}^{n \times n} \quad : \quad T_{ij} = \mathbb{P}(x_k = j \mid x_{k-1} = i)$$

whose element (i, j) yields the conditional transition probability that the Markov chain will be in state j at time k given that it has been in state i at time $k - 1$.

Lemma 2. *The transition matrix* $\mathbf{T}$ *has the following properties:*

$$T_{ij} \geq 0 \, \forall i,j$$
$$\sum_{j=1}^{n} T_{ij} = 1 \, \forall i$$

It follows that each row i of $\mathbf{T}$ is a probability distribution of the state found at the next time-step conditioned on i:

$$\mathbf{T}_{i*} = (T_{i1}, ..., T_{in}).$$

Generating realizations / trajectories Let us assume that in general the first state x_0 is drawn from an initial distribution $\mathbf{p}_0$. Then, a realization of length $N+1$ can be generated as follows:

1. Draw x_0 from the initial distribution $\mathbf{p}_0$
2. For $k = 0, ..., N-1$: draw x_{k+1} from the discrete distribution $[T_{x_k,1}, ..., T_{x_k,n}]$

Ensemble evolution The probability to find the chain at state i at time k, $p_{k,i}$, can be computed by considering all possible realizations from the previous step $k-1$:

$$\begin{aligned} p_{k,i} &= p_{k-1,1} T_{1i} + ... + p_{k-1,n} T_{ni} \\ &= \sum_{j=1}^{n} p_{k-1,j} T_{ji} \end{aligned}$$

Define the probability vector $\mathbf{p}_k = (p_{k,1}, ..., p_{k,n})^T$, this is compactly written as:

$$\mathbf{p}_k^T = \mathbf{p}_{k-1}^T \mathbf{T}.$$

Applying this equation k times starting from $\mathbf{p}_0$ yields the **Chapman-Kolmogorow equation**:

Lemma 3. *If the chain is started from an initial distribution $\mathbf{p}_0$, then the distribution at time step k is given by*

$$\mathbf{p}_k^T = \mathbf{p}_0^T \mathbf{T}^k.$$

where $\mathbf{T}^k$ is the kth power of matrix $\mathbf{T}$. Thus, the powers of $\mathbf{T}$ are still stochastic matrices, and serve as the propagators for longer timesteps:

$$(\mathbf{T}^k)_{ij} = \mathbb{P}(x_k = j \mid x_0 = i).$$

Connectedness of a chain

Definition 4. State j is **accessible** from state i (written $i \to j$), if and only if there exists a finite sequence of states

$$i = i_0,\ i_1,\ ...,\ i_{n-1},\ i_n = j$$

such that $T_{i_k,i_{k+1}} > 0$ for all $k \in \{0, 1, ..., n-1\}$. Thus, $i \to j$ if there is a nonzero probability that the Markov chain reaches j after a finite number of steps when starting from i.
If both, $i \to j$ and $j \to i$, then we say that i and j **communicate** (written $i \leftrightarrow j$).
A **communication class** $C \subseteq S$ is a set of states whose members communicate, i.e. $i \leftrightarrow j$ for all $i, j \in C$, , and no state in C communicates with any state not in C.
A finite Markov chain (or equivalently, its transition matrix $\mathbf{T}$) is **irreducible**, if it has a single communicating class $C = S$.

Example 5. The transition matrix

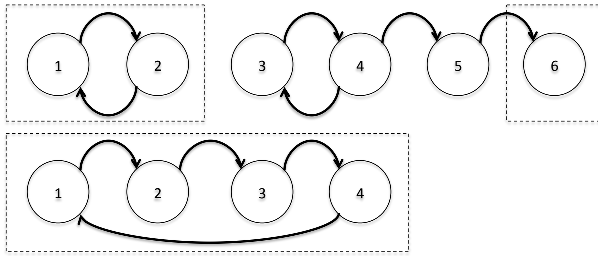
$$\mathbf{T} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.4 & 0.4 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

has the communication classes $\{1, 2\}$, $\{3, 4\}$, $\{5\}$, $\{6\}$. The communication class $\{3, 4\}$ is connected to $\{5\}$, and $\{5\}$ is connected to $\{6\}$, so $\{3, 4\}$ and $\{5\}$ are not closed. The only closed communication classes are $\{1, 2\}$ and $\{6\}$. $\mathbf{T}$ is reducible (not irreducible). On the other hand, the transition matrix

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

has the single closed communication class $S = \{1, 2, 3, 4\}$. $\mathbf{T}$ is thus irreducible.

Determination of Communication Classes In order to design an efficient algorithm to compute the communication classes, it is useful to view the transition matrix as a graph. We define the **connectivity graph** $D = (S, A)$. D is a **directed graph**, consisting of a set of **nodes** and **arrows** connecting these nodes. Here, D has the node set S, i.e. each node represents a state of the Markov chain. The arrow set A consists of all arrows that connect state i to j if and only if $T_{ij} > 0$ and $i \neq j$. The connectivity graphs of the two transition matrices above are:



We first introduce the **depth-first search** algorithm as an approach to traverse the nodes of a graph by following its arrows:

Algorithm 1 DFS(D, v, E): Pseudocode for depth-first search in a digraph D, staring from node v

---

Input: Digraph D, starting node v, List of explored nodes E.
Output: Updated E
Label node v as explored.
For all outgoing arrows $a = (v, w)$:
    If node w is unexplored then update E by DFS(D, w, E)
Append v to list of found nodes E.

Depth-first search starts traversing the graph at some specified starting node v and then returns the set of nodes E that are accessible from v.

Kosaraju's algorithm then uses depth-first search and exploits the fact that the transpose graph of D (the same graph with the direction of every arrow reversed) has exactly the same strongly connected components as D:

Algorithm 2 Kosaraju(D): Pseudocode of Kosaraju's strong component algorithm

---

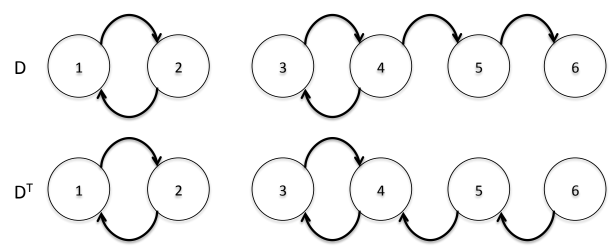
Input: Digraph D
Output: Set of communication classes, $\mathcal{C}$
Create empty list $V = ()$.
While V does not contain all nodes:
    Choose an arbitrary node v not in V.
    Update V by DFS (D, v, V)
Let D^T bet the transpose graph of D (directions of all arcs reversed)
While V is nonempty
    Let v be the last node in V
    Compute C by DFS (D^T, v, C) with C initially empty.
    C is the communication class containing v. Add C to $\mathcal{C}$.
    Remove all nodes in C from the graph D and the list V.

Indeed, C found in the second step of the second loop is a communication class: As we are traversing the transposed graph, we find all nodes that can access v, i.e. $v' \to v$ for all $v' \in C$. Suppose that one $v' \in C$ was not accessible from v. Then we would not have found v' in the last call of DFS starting from v. It follows that we would have found v' in an earlier run of DFS, but this implies that we would found v as well.

Consider the first example shown above. If we start with node 1, the DFS algorithm would first identify nodes $\{1, 2\}$, if we continue with node 3, then it would subsequently find $\{3, 4, 5, 6\}$. After the first stage, V would be given by:

$$V = (2, 1, 6, 5, 4, 3)$$

We now transpose the graph:



And call DFS starting from the last node in V, which is $v = 3$. We find the set $\{3, 4\}$. These nodes are removed from V and D^T. V is now:

$$V = (2, 1, 6, 5)$$

In the subsequent iterations we find the sets $\{5\}$, $\{6\}$, and finally $\{1, 2\}$. The algorithm ends with the communication classes:

$$\mathcal{C} = \{\{1, 2\}, \{3, 4\}, \{5\}, \{6\}\}.$$