Markov chains and applications

Dr. András Pongrácz

Week 12, University of Debrecen

Evolutionary processes

Involves partial differential equations, game theory, etc.

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Typically, the set of states is large, but each state has a low out-degree.

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Typically, the set of states is large, but each state has a low out-degree. Imagine the N-dimensional cube $\{0,1\}^N$, where in one step we change a coordinate, uniformly at random:

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Typically, the set of states is large, but each state has a low out-degree. Imagine the N-dimensional cube $\{0,1\}^N$, where in one step we change a coordinate, uniformly at random: 2^N states, each with out-degree N.

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Typically, the set of states is large, but each state has a low out-degree. Imagine the N-dimensional cube $\{0,1\}^N$, where in one step we change a coordinate, uniformly at random: 2^N states, each with out-degree N.

If an objective function is given, then the task is to find its maximum.

Involves partial differential equations, game theory, etc. However, a simplified discrete model can often be considered as an approximation.

Typically, the set of states is large, but each state has a low out-degree. Imagine the N-dimensional cube $\{0,1\}^N$, where in one step we change a coordinate, uniformly at random: 2^N states, each with out-degree N.

If an objective function is given, then the task is to find its maximum. A good idea is to make short random walks, looking for local improvement.

In game theory: mathematical models of economics.

Applications

In game theory : mathematical models of economics.

In biology : modeling actual evolution.

Applications

In game theory : mathematical models of economics.

In biology : modeling actual evolution.

In programming : evolutionary programs, improving themselves gradually (with or without human engineering).

Applications

In game theory : mathematical models of economics.

In biology : modeling actual evolution.

In programming : evolutionary programs, improving themselves gradually (with or without human engineering). Used to develop chess programs, antivirus softwares, etc.

Applications

In game theory : mathematical models of economics.

In biology : modeling actual evolution.

In programming : evolutionary programs, improving themselves gradually (with or without human engineering). Used to develop chess programs, antivirus softwares, etc. Often there are two competing programs evolving in parallel, enhancing each other (cop and robber).

Voting protocols

modeling (computer or real) viruses or gossip: Hedetniemi et al. (1988), Karp et al. (2000), Acan et al. (2014), etc.

Background and motivation

modeling (computer or real) viruses or gossip : Hedetniemi et al. (1988), Karp et al. (2000), Acan et al. (2014), etc.

synchronizing computers

modeling (computer or real) viruses or gossip: Hedetniemi et al. (1988), Karp et al. (2000), Acan et al. (2014), etc.

synchronizing computers

data sharing: Tran et al. (2004), Locher et al. (2007), Cigno et al. (2008), Russo (2009), etc.

Background and motivation

modeling (computer or real) viruses or gossip : Hedetniemi et al. (1988), Karp et al. (2000), Acan et al. (2014), etc.

synchronizing computers

data sharing : Tran et al. (2004), Locher et al. (2007), Cigno et al. (2008), Russo (2009), etc.

simulating the behavior of voters

Background and motivation

modeling (computer or real) viruses or gossip: Hedetniemi et al. (1988), Karp et al. (2000), Acan et al. (2014), etc.

synchronizing computers

data sharing: Tran et al. (2004), Locher et al. (2007), Cigno et al. (2008), Russo (2009), etc.

simulating the behavior of voters

social models: Holme, Newman (2006), Durrett et al. (2012), Basu, Sly (2015)

Donnelly, Welsh (1983) : continuous push and pull protocols

Background and motivation

Donnelly, Welsh (1983) : continuous push and pull protocols

Nakata (1999), Hassin, Peleg (2001) : discrete protocols

Background and motivation

Donnelly, Welsh (1983) : continuous push and pull protocols

Nakata (1999), Hassin, Peleg (2001) : discrete protocols

Oliveira (2012), Cooper et al. (2013) : connection with coalescence

Background and motivation

Donnelly, Welsh (1983) : continuous push and pull protocols

Nakata (1999), Hassin, Peleg (2001) : discrete protocols

Oliveira (2012), Cooper et al. (2013) : connection with coalescence

Cooper, Rivera (2015) : introduction of linear protocols

Background and motivation

Donnelly, Welsh (1983) : continuous push and pull protocols

Nakata (1999), Hassin, Peleg (2001) : discrete protocols

Oliveira (2012), Cooper et al. (2013) : connection with coalescence

Cooper, Rivera (2015) : introduction of linear protocols

Cooper et al. (2015) : introduction of discordant protocols

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

„Push protocol": randomly chosen vertex convinces a randomly chosen neighbor.

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

„Push protocol": randomly chosen vertex convinces a randomly chosen neighbor.

„Pull protocol": randomly chosen vertex is convinced by a randomly chosen neighbor.

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

„Push protocol”: randomly chosen vertex convinces a randomly chosen neighbor.

„Pull protocol”: randomly chosen vertex is convinced by a randomly chosen neighbor.

„Oblivious protocol”: randomly chosen edge, flip a coin whose opinion is accepted by both.

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

„Push protocol": randomly chosen vertex convinces a randomly chosen neighbor.

„Pull protocol": randomly chosen vertex is convinced by a randomly chosen neighbor.

„Oblivious protocol": randomly chosen edge, flip a coin whose opinion is accepted by both.

There are also synchronous protocols at use, where multiple vertices change their opinion at one turn.

Linear voting protocols

Given a finite, simple, connected graph $G = (V, E)$.

„Push protocol”: randomly chosen vertex convinces a randomly chosen neighbor.

„Pull protocol”: randomly chosen vertex is convinced by a randomly chosen neighbor.

„Oblivious protocol”: randomly chosen edge, flip a coin whose opinion is accepted by both.

There are also synchronous protocols at use, where multiple vertices change their opinion at one turn.

Common generalization: linear voting model.

Linear model

Sates : column vectors of length n (2^n of them), two absorbing states : constant vectors.

Linear model

Sates : column vectors of length n (2^n of them), two absorbing states : constant vectors.

$M_1, \ldots, M_k$: stochastic 0-1 matrices

Linear model

Sates : column vectors of length n (2^n of them), two absorbing states : constant vectors.

$M_1, \dots, M_k$: stochastic 0-1 matrices

$p_1, \dots, p_k$: a probability distribution

Linear model

Sates: column vectors of length n (2^n of them), two absorbing states: constant vectors.

$M_1, \ldots, M_k$: stochastic 0-1 matrices

$p_1, \ldots, p_k$: a probability distribution

In each round, we sample a matrix and multiply the vector of opinions from the left.

Linear model

Sates: column vectors of length n (2^n of them), two absorbing states: constant vectors.

$M_1, \ldots, M_k$: stochastic 0-1 matrices

$p_1, \ldots, p_k$: a probability distribution

In each round, we sample a matrix and multiply the vector of opinions from the left.

Average matrix: $M = p_1 M_1 + \cdots + p_k M_k$

Winning probabilities

Theorem

If M is ergodic, then the probability that the process ends in the consensus $\underline{1}$ provided that the initial state is $\underline{\xi}$ is $\underline{\mu}^* \underline{\xi}$, where $\underline{\mu}^*$ is the (unique) stationary distribution of M.

Winning probabilities

Theorem

If M is ergodic, then the probability that the process ends in the consensus $\underline{1}$ provided that the initial state is $\underline{\xi}$ is $\underline{\mu}^*\underline{\xi}$, where $\underline{\mu}^*$ is the (unique) stationary distribution of M.

Boundary conditions: $\underline{\mu}^*\underline{0} = 0, \underline{\mu}^*\underline{1} = 1$.

Winning probabilities

Theorem

If M is ergodic, then the probability that the process ends in the consensus $\underline{1}$ provided that the initial state is $\underline{\xi}$ is $\underline{\mu}^*\underline{\xi}$, where $\underline{\mu}^*$ is the (unique) stationary distribution of M.

Boundary conditions: $\underline{\mu}^*\underline{0} = 0, \underline{\mu}^*\underline{1} = 1$.

By the law of total probability:

Winning probabilities

Theorem

If M is ergodic, then the probability that the process ends in the consensus $\underline{1}$ provided that the initial state is $\underline{\xi}$ is $\underline{\mu}^*\underline{\xi}$, where $\underline{\mu}^*$ is the (unique) stationary distribution of M.

Boundary conditions: $\underline{\mu}^*\underline{0} = 0, \underline{\mu}^*\underline{1} = 1$.

By the law of total probability:

$$\sum p_i \underline{\mu}^* M_i \underline{\xi} = \underline{\mu}^* (\sum p_i M_i) \underline{\xi} = \underline{\mu}^* M \underline{\xi} = \underline{\mu}^* \underline{\xi}.$$

Winning probabilities

Theorem

If M is ergodic, then the probability that the process ends in the consensus $\underline{1}$ provided that the initial state is $\underline{\xi}$ is $\underline{\mu}^*\underline{\xi}$, where $\underline{\mu}^*$ is the (unique) stationary distribution of M.

Boundary conditions: $\underline{\mu}^*\underline{0} = 0, \underline{\mu}^*\underline{1} = 1$.

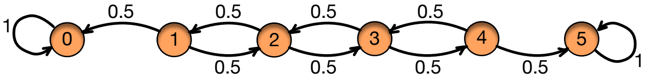
By the law of total probability:

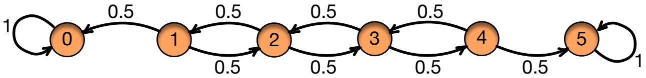
$\sum p_i \underline{\mu}^* M_i \underline{\xi} = \underline{\mu}^* (\sum p_i M_i) \underline{\xi} = \underline{\mu}^* M \underline{\xi} = \underline{\mu}^* \underline{\xi}$.

The runtime can be estimated too, e.g., by the conductance of the graph or the coalescence time, providing polynomial upper bounds.

Gambler's ruin

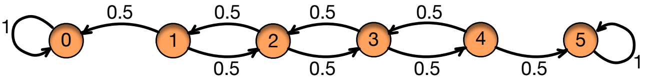
Gambler's ruin (a.k.a. drunkard's walk)





$$Q = \begin{pmatrix} 0 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} \Rightarrow NR = \begin{pmatrix} 4/5 & 1/5 \\ 3/5 & 2/5 \\ 2/5 & 3/5 \\ 1/5 & 4/5 \end{pmatrix} \quad N\underline{\mathbf{1}} = \begin{pmatrix} 4 \\ 6 \\ 6 \\ 4 \end{pmatrix}$$

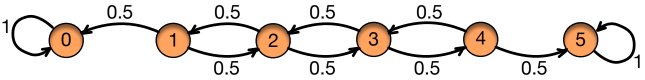
Gambler's ruin (a.k.a. drunkard's walk)



$$Q = \begin{pmatrix} 0 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} \Rightarrow NR = \begin{pmatrix} 4/5 & 1/5 \\ 3/5 & 2/5 \\ 2/5 & 3/5 \\ 1/5 & 4/5 \end{pmatrix} \quad N\underline{\mathbf{1}} = \begin{pmatrix} 4 \\ 6 \\ 6 \\ 4 \end{pmatrix}$$

In general: probability to be absorbed at the right-most state is k/n, and the expected runtime is $k(n-k)$.

Gambler's ruin (a.k.a. drunkard's walk)



$$Q = \begin{pmatrix} 0 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0 \end{pmatrix} \Rightarrow NR = \begin{pmatrix} 4/5 & 1/5 \\ 3/5 & 2/5 \\ 2/5 & 3/5 \\ 1/5 & 4/5 \end{pmatrix} \quad N\underline{\mathbf{1}} = \begin{pmatrix} 4 \\ 6 \\ 6 \\ 4 \end{pmatrix}$$

In general: probability to be absorbed at the right-most state is k/n, and the expected runtime is $k(n-k)$. It is possible to compute the fundamental matrix parametrically in general.

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1).

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not: when an opinion spreads, the probabilities are lopsided.

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not : when an opinion spreads, the probabilities are lopsided.

Discordant versions : we only pick uniformly at random from those who disagree.

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not: when an opinion spreads, the probabilities are lopsided.

Discordant versions: we only pick uniformly at random from those who disagree. (In case of oblivious, we sample from the discordant edges.)

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not : when an opinion spreads, the probabilities are lopsided.

Discordant versions : we only pick uniformly at random from those who disagree. (In case of oblivious, we sample from the discordant edges.)

Makes sense in practice : no idle rounds.

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not : when an opinion spreads, the probabilities are lopsided.

Discordant versions : we only pick uniformly at random from those who disagree. (In case of oblivious, we sample from the discordant edges.)

Makes sense in practice : no idle rounds.

On cycle graphs, the three discordant protocols are very similar (all very close to a gambler's ruin), so the game is nearly fair and concludes quickly.

Discordant protocols

The oblivious protocol is a gambler's ruin (the number of vertices with opinion 1). The others are not : when an opinion spreads, the probabilities are lopsided.

Discordant versions : we only pick uniformly at random from those who disagree. (In case of oblivious, we sample from the discordant edges.)

Makes sense in practice : no idle rounds.

On cycle graphs, the three discordant protocols are very similar (all very close to a gambler's ruin), so the game is nearly fair and concludes quickly. The discordant push (or some variant) is often used in computer science in P2P protocols (data sharing and synchronizing computers).

Exercises

Exercises

1. Compute the fundamental matrix of the drunkard walk (fair gambler's ruin).

Exercises

1. Compute the fundamental matrix of the drunkard walk (fair gambler's ruin).
2. Compute the first four moments of the gambler's ruin.

Exercises

1. Compute the fundamental matrix of the drunkard walk (fair gambler's ruin).
2. Compute the first four moments of the gambler's ruin.
3. Find and verify the formulas for the probabilities of absorption and the expected runtime in the unfair gambler's ruin problem, when the probability to move to the left in each transient state is a fixed $p > 1/2$. (And understand why it is important to shuffle the deck properly before a new game of blackjack or poker.)

Exercises

1. Compute the fundamental matrix of the drunkard walk (fair gambler's ruin).
2. Compute the first four moments of the gambler's ruin.
3. Find and verify the formulas for the probabilities of absorption and the expected runtime in the unfair gambler's ruin problem, when the probability to move to the left in each transient state is a fixed $p > 1/2$. (And understand why it is important to shuffle the deck properly before a new game of blackjack or poker.)
4. Show that the fair gambler's ruin is indeed a fair game, i.e., a martingale, whereas the unfair gambler's ruin in the previous problem is a supermartingale.