IK 2018

some excerpts from:

Dynamical Systems: a Navigation Guide

Herbert Jaeger

What is a dynamical system?

A DS is any real or artificial or formal system that evolves over time.

It is impossible not to be a dynamical system! Because, if you *are*, you are *being in time*.

Examples

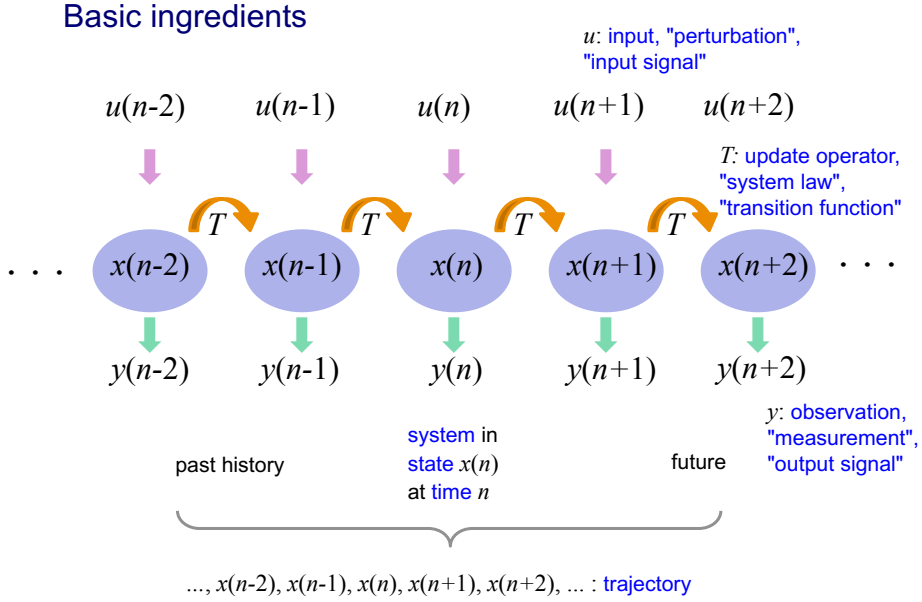
- A water molecule, a waterdrop, a river, an ocean
- The Universe
- A calcium channel, a synapse, a dendrite, a neuron, a microcircuit, ... a brain, a nervous system, a body
- Life on earth
- A bitstream, a network of communicating signal sources, a language generating program, a society of linguistic agents
- Mathematics (as a growing body of theorems and proofs)
- You
- What you think about you

In sum:

- Everything that is not dead or boring.

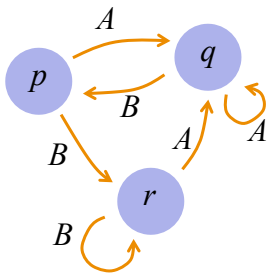
There isn't and there can't be a universal theory of dynamical systems.
We have to face a diversity of methods.

Basic ingredients

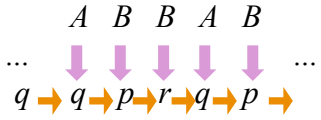


Part 2: A Zoo of Finite-State Models

Deterministic finite-state automata (DFA)



example trajectory:



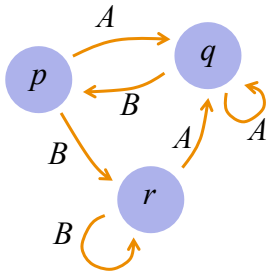
A DFA is defined by:

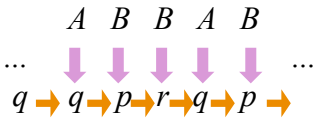
- a finite set of states, e.g. $Q = \{p, q, r\}$
- a finite set of input symbols, e.g. $\Sigma = \{A, B\}$
- a transition function T: $Q \times \Sigma \rightarrow Q$, can be written as table, e.g.

	A	B
p	q	r
q	q	p
r	q	r

- A DFA defines input-sequence dependent state trajectories

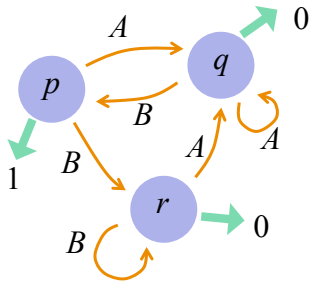
DFAs, comments



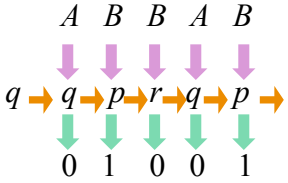


- deterministic
- can be used e.g. for modeling ion channel states or an agent's model of how the world Q reacts on agent's actions Σ
- simplicity is deceptive: a PC can be considered a DFA, with more-than-astronomical-sized (but finite) $Q = \{$`all possible logic gate state combinations`$\}$
- states are "fully observable"
- inferring a DFA from observed trajectories is easy
- DFAs are standard tool for theoretical CS, then used only for finite sequences ("words")

Moore- and Mealy-Machines



example Moore trajectory:



A Moore Machine is a DFA, equipped additionally with

- a finite set of output symbols, e.g. $O = \{0, 1\}$
- a translation (observation) function $\rho: Q \to O$

A Mealy Machine is similar, except the observations are "emitted" from transition arrows: $\rho: Q \times \Sigma \to O$

Both are deterministic.

Efficient methods to infer Moore / Mealy machines from input-output data are known.

Literature

There are many textbooks covering finite automata – they form a core part of theoretical CS and any theoretical CS textbook will cover them (among other topics). A classic is

Hopcroft, John E. , Motwani, Rajeev, and Ullman, Jeffrey: *Introduction to Automata Theory*, 2nd edition. Addison-Wesley, 2001

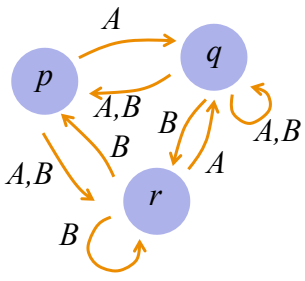
Textbook concentrating on finite automata which covers both the use of automata for finite-word languages and for infinite-sequence languages

A. de Vries: Finite Automata: Behavior and Synthesis. Elsevier, 2014

State-of-the-art entrance paper for learning Mealy machines from data:

Steffen, B., Howar, F., & Merten, M. (2011). Introduction to active automata learning from a practical perspective. In *Formal Methods for Eternal Networked Software Systems* (pp. 256-296). Springer Berlin Heidelberg. http://ls5-www.cs.tu-dortmund.de/cms/en/research/papers/introduction-to-automata-learning-sfm2011.pdf

Non-deterministic finite-state automata (NFA)



example trajectories:

$A\ B\ B\ A\ B$

... $q \to q \to p \to r \to q \to p \to$ ...

or $q \to p \to r \to r \to q \to r \to$

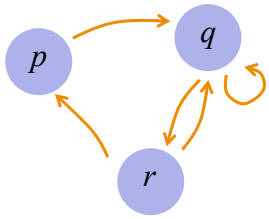
A NFA is defined by:

- a finite set of states, e.g. $Q = \{p, q, r\}$
- a finite set of input symbols, e.g. $\Sigma = \{A, B\}$
- a transition function T: $Q \times \Sigma \to \mathrm{Pot}(Q)$, (Pot: power set), e.g.

	A	B
p	$\{q, r\}$	$\{r\}$
q	$\{p, q\}$	$\{p, q, r\}$
r	$\{q\}$	$\{p, r\}$

NFAs, comments

special case: no input symbols

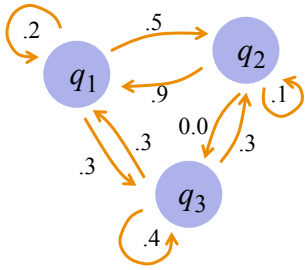


example trajectory:

$q \to q \to r \to p \to q \to r \to$

- an NFA can yield several (typically, infinitely many) different state sequences on given input sequence
- no probabilities involved; a given state sequence cannot be said to be "more probable" than another
- this is called a non-deterministic system, as opposed to "deterministic" and to "stochastic"
- nondeterministic models capture what is possible vs. what is impossible to observe
- special case: no input (or equivalently, one-element input set)

Finite-dimensional Markov chains



$$\boldsymbol{p} = \begin{pmatrix} 0.5 \\ 0.0 \\ 0.5 \end{pmatrix}$$

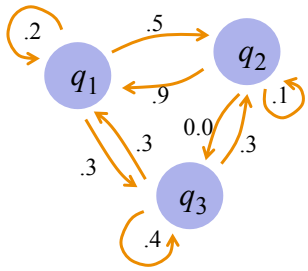
A (finite-dimensional) Markov chain (MC) is defined by:

- a finite set of states, e.g. $Q = \{q_1, q_2, q_3\}$
- an initial state distribution $\boldsymbol{p} \in Prob(Q)$, where $Prob(Q)$ is the set of probability distributions over Q
- a transition kernel T: $Q \to Prob(Q)$
- T can be written as stochastic transition matrix ("Markov matrix"), e.g.

	q_1	q_2	q_3
q_1	0.2	0.5	0.3
q_2	0.9	0.1	0.0
q_3	0.3	0.3	0.4

rows sum to 1

Finite-dimensional MCs, comments 1



$$\boldsymbol{p} = \begin{pmatrix} 0.5 \\ 0.0 \\ 0.5 \end{pmatrix}$$

- used to describe trajectories that start at time $n = 0$
- probability that a trajectory starts with

  $q_{i0}, q_{i1}, \ldots, q_{in}$

  is

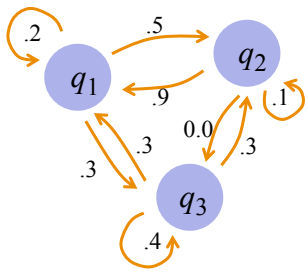
$$P(X_0 = q_{i_0},\ldots, X_n = q_{i_n}) =$$
$$= P(X_0 = q_{i_0}) \cdot P(X_1 = q_{i_1} \mid X_0 = q_{i_0}) \cdot \ldots \cdot$$
$$\cdot P(X_n = q_{i_n} \mid X_{n-1} = q_{i_{n-1}})$$
$$= P(X_0 = q_{i_0}) \cdot \prod_{k=1}^{n} P(X_k = q_{i_k} \mid X_{k-1} = q_{i_{k-1}})$$
$$= \boldsymbol{p}(i_0)\, T(i_0, i_1) \ldots T(i_{n-1}, i_n)$$

- a MC specifies a stochastic process

  $(X_n)_{n=0,1,2,\ldots}$

  with values in Q

Finite-dimensional MCs, comments 2



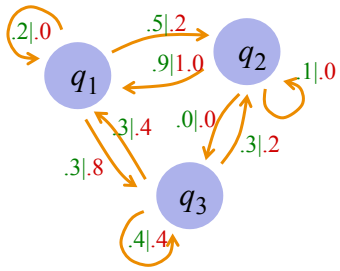
$$\boldsymbol{p} = \begin{pmatrix} 0.5 \\ 0.0 \\ 0.5 \end{pmatrix}$$

- the crucial defining property to make a finite-valued stochastic process a Markov chain: the Markov property

$$P(X_{n+1} = q_{i_{n+1}} \mid X_0 = q_{i_0},\ldots, X_n = q_{i_n}) =$$
$$= P(X_{n+1} = q_{i_{n+1}} \mid X_n = q_{i_n})$$

- What is going to happen next (probabilities to observe q_{in+1}) only depends on current state q_{in}, not on previous state history
- MCs are "memoryless" systems

Controlled Markov chains



$$\boldsymbol{p} = \begin{pmatrix} 0.5 \\ 0.0 \\ 0.5 \end{pmatrix}$$

In a controlled MC the transition probabilities are switched by inputs. Components:

- a finite set of states, e.g. $Q = \{q_1, q_2, q_3\}$
- a finite set of inputs ("control actions"), e.g. $A = \{a_1, a_2\}$
- an initial state distribution $\boldsymbol{p} \in Prob(Q)$
- for each $a \in A$, a transition kernel $T_a: Q \to Prob(Q)$

T_{a_1}	q_1	q_2	q_3
q_1	0.2	0.5	0.3
q_2	0.9	0.1	0.0
q_3	0.3	0.3	0.4

T_{a_2}	q_1	q_2	q_3
q_1	0.0	0.2	0.8
q_2	1.0	0.0	0.0
q_3	0.4	0.2	0.4

- Update mechanism: switch transition kernel according to current input symbol

Literature

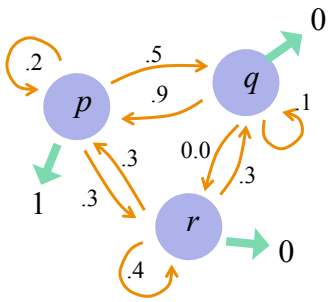
A classical monograph on controlled stochastic processes (general rigorous mathematical theory, not restricted to controlled MCs):

Gihman, I.I. and Skorohod, A.V., Controlled Stochastic Processes. Springer Verlag 1979

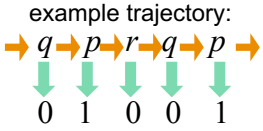
Controlling stochastic systems is, of course, also of prime importance in control engineering. In this field, the controlled systems often are systems that emit observable output, which is then included in the analysis and methods. A textbook:

R. F. Stengel, Stochastic optimal control: theory and application. John Wiley and Sons, 1986

Hidden Markov models (HMMs)



example trajectory:



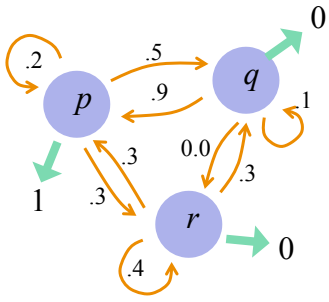
Defining components:

- a finite set of states, e.g. $Q = \{p, q, r\}$
- a finite set of outputs ("observables", "visibles"), e.g. $O = \{0, 1\}$
- an initial state distribution $\boldsymbol{p} \in Prob(Q)$
- a transition kernel $T : Q \to Prob(Q)$
- for every state $q \in Q$ and observable $o \in O$, an emission probability $P(o|q)$ to observe o when the hidden Markov state trajectory passes through q

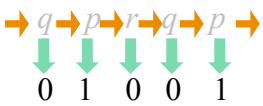
  or, equivalently,

  an emission function $\rho : Q \to O$ (as shown in example)

HMMs, comments 1



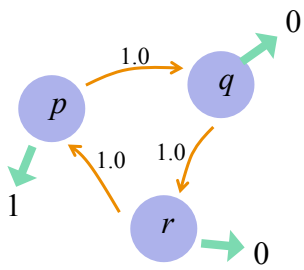
unobservable states



visible measurements

- widely and naturally applicable, because an experimenter often can't directly observe states q, only make measurements o of them
- available experimental data are only trajectories of observables, states are unobservable
- model inference task: from (empirical) measurement data (e.g., 0 1 0 0 1) infer underlying stochastic state transition system, that is...
- ... explain data by generative mechanism
- Example: Q = "brain states", O = "uttered phonemes"
- Example: Q = "state of a neuron", O = "spike"

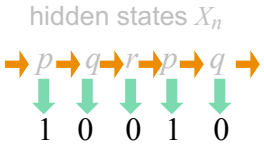
HMMs, comments 2



- The visible trajectories (values of random variables Y_n) "have memory":

$$P(Y_{n+1}=1 \mid Y_n=0) = 0.5$$
$$P(Y_{n+1}=1 \mid Y_n=0, Y_{n-1}=0) = 1.0$$

- The observables $(Y_n)_{n=0,1,2,\ldots}$ form a stochastic process in their own right, but this process does not have the Markov property

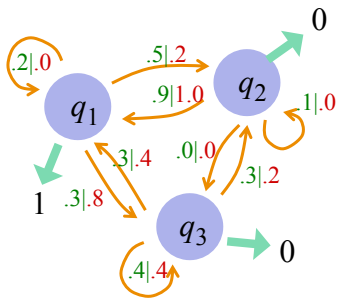


Literature

The classical tutorial text on HMMs, very readable (20000+ Google cites, boosted the popularity of HMMs in speech processing):

Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE* 77.2 (1989): 257-286. (many online copies)

Controlled hidden Markov models, aka POMDPs



- crossover of controlled MCs and HMMs
- also (widely) known as Partially Observable Markov Decision Processes (POMDPs)
- a basic tool in theory of autonomous agents / robotics / reinforcement learning in the machine learning sense
- In that context, a POMDP constitutes the agent's world model:
  - Q: external world states
  - A: agent's actions in world
  - O: sensory feedback from world
- methods available for learning a POMDP from A-O (action – sensor-feedback) timeseries data
- seems a natural model class to me also for neural dynamics and animal behavior

Literature

My favorite tutorial text on POMDPs, set in a context of agent learning and reinforcement learning:

Kaelbling, L.P., Littman, M.L., Cassandra, A.R., Planning and acting in partially observable stochastic domains. Artificial Intelligence 101 (1998), 99-134