Reinforcement Learning

Prof. Christian Bauckhage

lecture 05
from state machines to graphical models

in this “transitional” lecture, we look at *state transition models* of agent behavior

we recall the notions of *deterministic finite state machines*, *deterministic finite state transducers*, *non-deterministic finite state machines*, and *probabilistic finite state machines*

we will see that the latter can be understood as *Markov chains* which we will study in more detail later on

to prepare ourselves for that, we finally look at *Bayesian networks* / *graphical models* and how to use them for inference

outline

recap / introduction

so far, we looked at

- two-player, turn based, perfect information games
- state spaces and/or graphs (game trees, networks)
- tree search for decision making (minmax, MCTS)
- tree search for route planning (Dijkstra, A^*)

$\Leftrightarrow$ so far, we saw “classical” problem solving and decision making

next, we will look at

- *behavior programming*
- *behavioral inference*

questions

what's the big deal ?

isn't a "behavior" just a path in a search tree ?

isn't a "behavior" just a sequence of states ?

questions

what's the big deal ?

isn't a "behavior" just a path in a search tree ?

isn't a "behavior" just a sequence of states ?

answer

yes, but . . .

when we talked about games, we introduced

- configuration (instance of entities and attributes)
- game entities (e.g. game agents, weapons, . . .)
- game entity attributes (e.g. location, health, . . .)
- game state (configuration compliant with rules)

now, consider this

- modern video games have very large state spaces
- real world environments have even bigger state spaces

$\Rightarrow$ to master those, we may need a different perspective on “behavior”

behavior

- $\Leftrightarrow$ range of actions made by systems (natural or artificial) with regard to themselves and/or the environment
- $\Leftrightarrow$ response of systems to stimuli / inputs (internal or external)

disclaimer

from now on

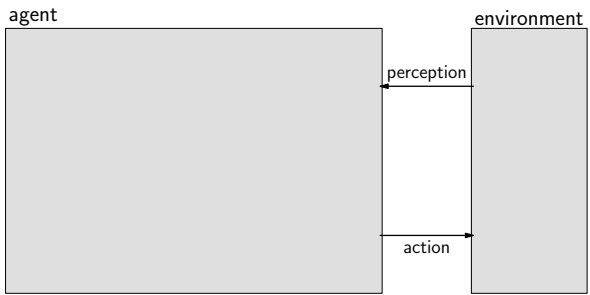
system $\equiv$ (artificial) agent

and (to keep our exposition simple)

behavior $\equiv$ individual agent behavior (as opposed to multi-agent (team) behavior)

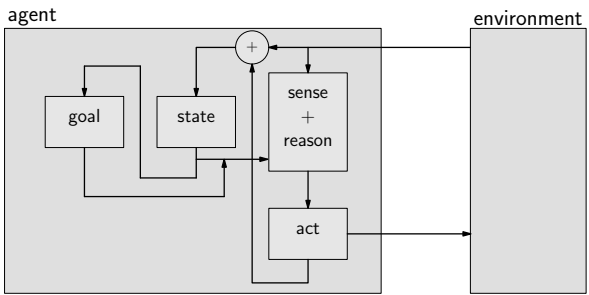
(artificial) agent

here is a diagram / classic idea



$\Leftrightarrow$ the agent is a “black box” and “separated” from the environment

since it's not that simple, here is a refined diagram



note

this doesn't really help and grows quickly out of hand

there actually is a discipline called "diagrammatology"

> *diagrammatology investigates the role of diagrams for thought and knowledge*

and the fact that the role of diagrams for thought and knowledge needs scientific study might be reason for concern ;-)

$\Leftrightarrow$ even if people can draw fancy diagrams, they may not understand whatever complex phenomenon their diagrams supposedly explain

question

why is it so difficult to model an agent ?

answer

let's see ...

the behavioral hierarchy

strategic behaviors
long term goals
(secure base, win game, . . .)

plan way ahead

tactical behaviors
short term goals
(set trap, protect flag carrier, . . .)

plan on the spot

reactive behaviors
immediate actions
(fire on sight, duck, seek cover, . . .)

no planning at all
$\Leftrightarrow$ *control problem*

state machine models

finite state machines or *automata* are a venerable model class in computer science

> W. McCulloch and W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics, 5, **1943**
>
> G.H. Mealy, *A Method for Synthesizing Sequential Circuits*, Bell System Technical J., 34(5), **1955**
>
> E.F. Moore, *Gedanken-experiments on Sequential Machines*, Automata Studies, vol. 34, **1956**

to this day, they are *the* baseline technique for modeling / programming the behavior of artificial agents such as NPCs in computer games (first done in Pac-Man, **1980**)

we therefore recall . . .

deterministic finite state machine (FSM)
or deterministic finite automaton (DFA)

$\Leftrightarrow$ a tuple

$$\boldsymbol{M} = \left(\mathcal{S}, \Sigma, \delta, S_0, \mathcal{F}\right)$$

where

$$\begin{aligned}
\mathcal{S} &\equiv \text{finite set of states} \\
\Sigma &\equiv \text{alphabet of inputs} \\
\delta : \mathcal{S} \times \Sigma \to \mathcal{S} &\equiv \text{transition function} \\
S_0 \in \mathcal{S} &\equiv \text{initial state} \\
\mathcal{F} \subseteq \mathcal{S} &\equiv \text{set of final states}
\end{aligned}$$

the above definition is the canonical one most commonly found in text books

the “alphabet of inputs Σ” occurs because of the pivotal role of state machines and automata in *formal language theory*

> N. Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, 2(2), **1959**
>
> J.E. Hopcroft and J.D. Ullman, *An Introduction to Automata Theory, Languages, and Computation*, **1979**

to tie this definition closer to the content of this course, we restate it like this . . .

deterministic finite state machine (FSM)

$\Leftrightarrow$ a tuple

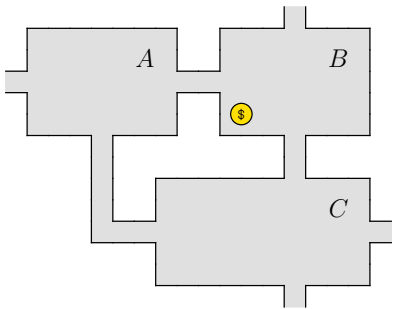
$$\boldsymbol{M} = \left(\mathcal{S}, \mathcal{A}, \delta, S_0, \mathcal{F}\right)$$

where

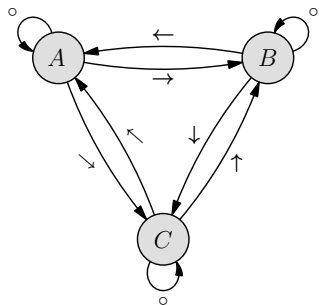
$$\begin{aligned}
\mathcal{S} &\equiv \text{finite set of states} \\
\mathcal{A} &\equiv \text{finite set of actions} \\
\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S} &\equiv \text{transition function} \\
S_0 \in \mathcal{S} &\equiv \text{initial state} \\
\mathcal{F} \subseteq \mathcal{S} &\equiv \text{set of final states}
\end{aligned}$$

example

recall that we have already seen such FSMs in lecture 01 where we looked at a map of three rooms and alleyways in a dungeon crawler game



an environment



reasonable state transitions

in this example, we have

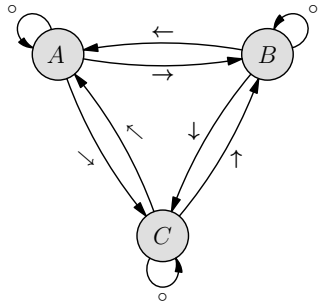
$$\mathcal{S} = \{A, B, C\}$$
$$\mathcal{A} = \{\circ, \rightarrow, \leftarrow, \downarrow, \uparrow, \searrow, \nwarrow\}$$

and the transition function

$$\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$$

is *defined* in terms of this *table*

δ	$\circ$	$\rightarrow$	$\leftarrow$	$\downarrow$	$\uparrow$	$\searrow$	$\nwarrow$
A	A	B	$-$	$-$	$-$	C	$-$
B	B	$-$	A	C	$-$	$-$	$-$
C	C	$-$	$-$	$-$	B	$-$	A



important observations

for finite $\mathcal{S}$ and $\mathcal{A}$, we can always conceptualize δ as a (lookup) table*

it is common that δ is a partial function

$\Leftrightarrow$ there may be pairs $(s_i, a_j) \in \mathcal{S} \times \mathcal{A}$ for which $\delta(s_i, a_j)$ is undefined

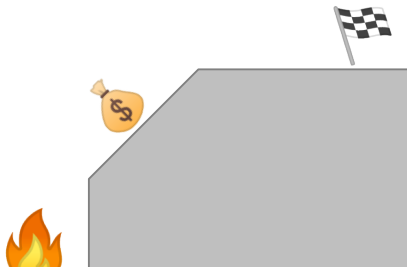
our example neither involves an initial state $S_0 \in \mathcal{S}$ nor a set $\mathcal{F} \subseteq \mathcal{S}$ of final states**

*in most interesting settings, this table will be huge / unworkable

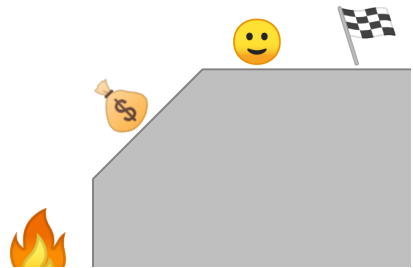
**to grasp the role of the latter, let us look at another example ...

example (inspired by Elliot Waite)

a super simple game environment



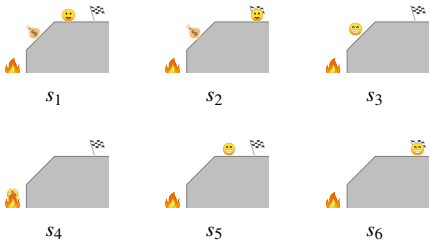
whose initial game state is



set of states

$$\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$$

where



set of actions

$$\mathcal{A} = \{\leftarrow, \circ, \rightarrow\}$$

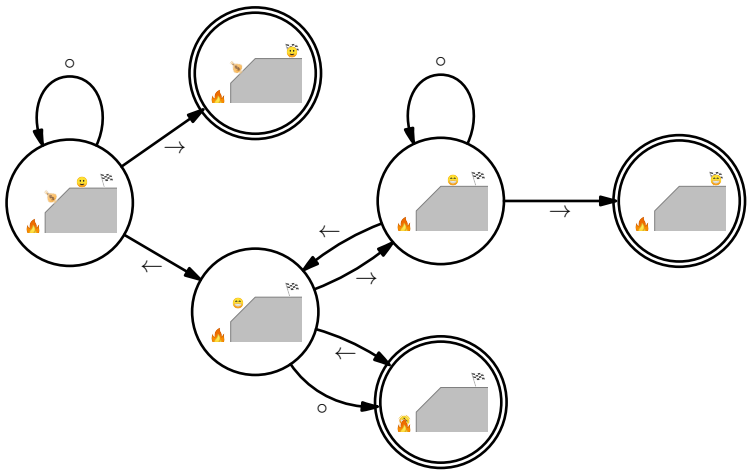
where

$\leftarrow$: go left

$\circ$: stay put

$\rightarrow$: go right

$\Rightarrow$ FSM model with $S_0 = s_1$ and $\mathcal{F} = \{s_2, s_4, s_6\}$



dsiclaimer

what we do next may seem strange . . .

we will consider the idea that state transitions are caused by *observations* rather than by *actions*

in the context of reinforcement learning, this is unusual but, as we will see, in the context of game AI, it is actually reasonable

to be honest, our real reason for considering this alternative view is that it allows us to introduce terminology which we will need next time . . .

event / observation driven FSM

$\Leftrightarrow$ a tuple

$$\boldsymbol{M} = \left(\mathcal{S}, \mathcal{O}, \delta, S_0, \mathcal{F}\right)$$

where

$$\begin{aligned}
\mathcal{S} &\equiv \text{finite set of states} \\
\mathcal{O} &\equiv \text{set of observations} \\
\delta : \mathcal{S} \times \mathcal{O} \to \mathcal{S} &\equiv \text{transition function} \\
S_0 \in \mathcal{S} &\equiv \text{initial state} \\
\mathcal{F} \subseteq \mathcal{S} &\equiv \text{set of final states}
\end{aligned}$$

example

we may represent the state of our simple game environment using a 2D observation vector

$$\boldsymbol{o} = \begin{bmatrix} l \\ x \end{bmatrix}$$

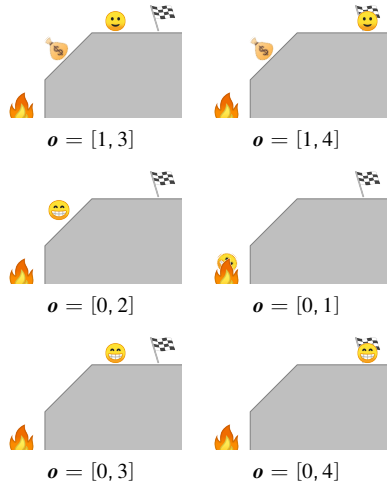
where

$$l \in \{0, 1\}$$

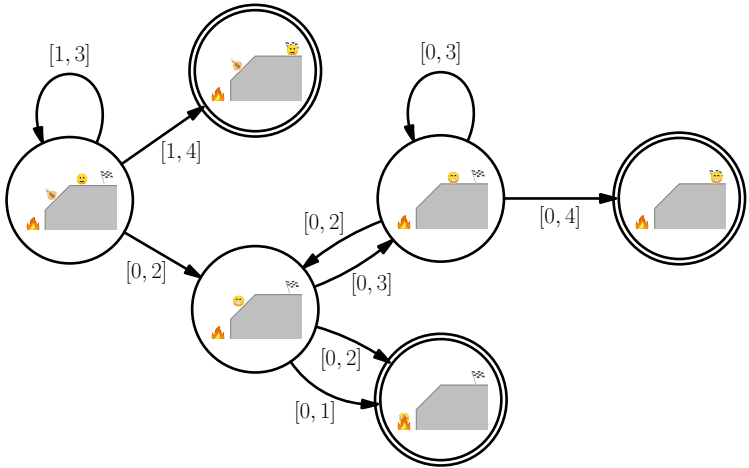
is a flag indicating if there is *loot* and

$$x \in \{1, 2, 3, 4\}$$

indicates the current player *position*



$\Rightarrow$ we could use the following observation driven FSM in a graphics engine that renders our game environment



remarks

if we were to use state machines to model or update game states,
we would have to pay attention to the following

for each state in the above model, there would likely be scripts that generate / manage *micro behavior*

$\Leftrightarrow$ states are accompanied by scripts that update attributes, render transitional animations, etc.

$\Leftrightarrow$ in the context of games, state machines usually produce output

$\Rightarrow$ **seen from this point of view, we should better consider *transducers***

deterministic finite state transducer (FST)

$\Leftrightarrow$ an FSM with output

$$\boldsymbol{T} = \left(\mathcal{S}, \mathcal{O}, \mathcal{Q}, \delta, \omega, S_0\right)$$

where $\mathcal{S}, \mathcal{O}, \delta, S_0$ as above and

$$\mathcal{Q} \equiv \text{set of outputs}$$

$$\omega : \mathcal{S} \to \mathcal{Q} \quad \text{(Moore output)}$$

$$\omega : \mathcal{S} \times \mathcal{O} \to \mathcal{Q} \quad \text{(Mealy output)}$$

remarks

in this RL course, we will not look under that rock !

however, for the fun of it, we mention the following . . .

since we mocked “overeager” diagrams in the introduction, we note that FSMs for real world settings may be large and complex

$\Rightarrow$ we might want to *minimize FSMs*

in fact, for every FSM, there exists a unique *minimal FSM*

two kinds of states can always be removed and/or merged

- unreachable states
- indistinguishable / equivalent states

for what follows, we need a few more definitions

$\mathcal{O}^\star \equiv$ set of all words over $\mathcal{O}$ $\quad (\Leftrightarrow$ Kleene star of $\mathcal{O})$

$\epsilon \in \mathcal{O}^\star \equiv$ empty sequence over $\mathcal{O}$

$w = o : os \in \mathcal{O}^\star \equiv$ word where $o \in \mathcal{O},\ os \in \mathcal{O}^\star$

$\delta^\star : \mathcal{S} \times \mathcal{O}^\star \to \mathcal{S} \equiv$ transition function such that

$$\delta^\star\big(s, o : os\big) = \begin{cases} \delta\big(s, o\big) & \text{if } os = \epsilon \\ \delta^\star\Big(\delta\big(s, o\big), os\Big) & \text{otherwise} \end{cases}$$

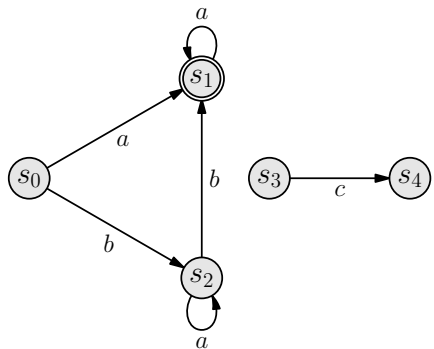
unreachable states

$\Leftrightarrow$ a state s_i is unreachable if there is no word

$$w \in \mathcal{O}^\star$$

such that

$$s_i = \delta^\star\left(S_0, w\right)$$

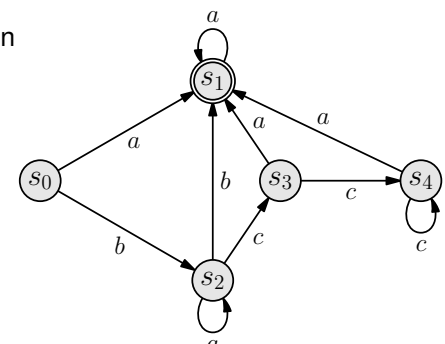


equivalent states

$\Leftrightarrow$ two states s_i and s_j are equivalent $s_i \sim s_j$, if

$$\forall\, w \in \mathcal{O}^\star : \delta^\star\big(s_i, w\big) \in \mathcal{F} \Leftrightarrow \delta^\star\big(s_j, w\big) \in \mathcal{F}$$

$\Leftrightarrow$ the machine does the same when started in either of the two states



note

there are efficient algorithms $\big(O\big(n \log n\big), n = |\mathcal{S}|\big)$
which identify equivalent states (Hopcroft, **1971**)

$\Rightarrow$ DFA / FSM minimization is solved and allows for
optimizing representations of behavior

there are numerous generalizations of this kind
of state-based behavior representations

a (theoretically) important one is the following ...

disclaimer

we now switch back from observations $\mathcal{O}$ to actions $\mathcal{A}$

non-deterministic automaton / state machine (NFA / NFSM)

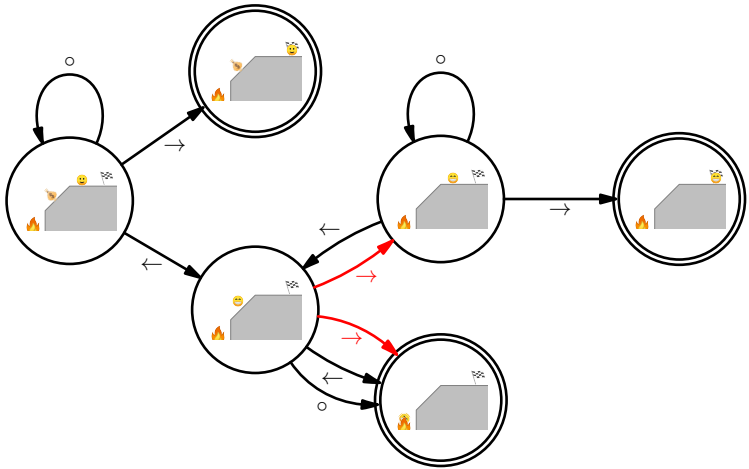
$\Leftrightarrow$ an FSM where δ is replaced by

$$\Delta : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$$

$\Leftrightarrow$ when executing a in state s, the machine may transit to one out of several states

example

here is an NSFM model for our simple game



question

how can this be ?

answer

(game) *environments may be adversarial, antagonistic, or non-deterministic*

if you go right in state s_3, you may succeed and reach state s_5 just as intended

if you go right in state s_3, you may also slip and find yourself in state s_4 instead

note

“one can show” that NFAs are equivalent to DFAs

we can generalize the whole approach even further

note: what follows is essential . . .

probabilistic automaton / state machine

$\Leftrightarrow$ a generalization of NFSMs

note

instead of a function

$$\Delta : \mathcal{S} \times \mathcal{A} \to 2^{\mathcal{S}}$$

we may think of a relation

$$\Delta \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$$

example

function

$$
\begin{aligned}
\Delta(s_1, \circ) &= \{s_1\} \\
\Delta(s_1, \rightarrow) &= \{s_2\} \\
\Delta(s_1, \leftarrow) &= \{s_3\} \\
\Delta(s_3, \circ) &= \{s_4\} \\
\Delta(s_3, \rightarrow) &= \{s_4, s_5\} \\
\Delta(s_3, \leftarrow) &= \{s_4\} \\
&\vdots
\end{aligned}
$$

relation

$$
\begin{aligned}
\Delta = \Big\{ &(s_1, \circ, s_1), (s_1, \rightarrow, s_2), (s_1, \rightarrow, s_3), \\
&(s_3, \circ, s_4), (s_3, \rightarrow, s_4), (s_3, \rightarrow, s_5), (s_3, \leftarrow, s_4), \ldots \Big\}
\end{aligned}
$$

note

instead of a relation

$$\Delta \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$$

we may think of an indicator function

$$\tau : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \{0, 1\}$$

where

$$\tau(s, a, s') = \begin{cases} 1 & \text{if } (s, a, s') \in \Delta \\ 0 & \text{otherwise} \end{cases}$$

note

$\Rightarrow$ for every $a \in \mathcal{A}$, we may define a (transition) matrix $\boldsymbol{M}_a$ with

$$\left[\boldsymbol{M}_a\right]_{s's} = \tau\left(s, a, s'\right) \in \left\{0, 1\right\}$$

note

$\Rightarrow$ for every $a \in \mathcal{A}$, we may define a (transition) matrix $\boldsymbol{M}_a$ with

$$\left[\boldsymbol{M}_a\right]_{s's} = \tau\left(s, a, s'\right) \in \left\{0, 1\right\}$$

we may replace $\boldsymbol{M}_a$ by a *column stochastic matrix* $\boldsymbol{P}_a$ where

$$\left[\boldsymbol{P}_a\right]_{s's} \geqslant 0$$

$$\sum_{s'} \left[\boldsymbol{P}_a\right]_{s's} = 1$$

note

$\Rightarrow$ states of a probabilistic state machine are *stochastic vectors* $\boldsymbol{v} \in \mathbb{R}^{|S|}$ where

$$v_s \geqslant 0$$

$$\sum_s v_s = 1$$

note

$\Rightarrow$ states of a probabilistic state machine are *stochastic vectors* $\boldsymbol{v} \in \mathbb{R}^{|S|}$ where

$$v_s \geqslant 0$$

$$\sum_s v_s = 1$$

for instance, when starting in state $\boldsymbol{v}$ and performing actions $as = a_3 : a_1 : a_5$, the machine will be in state

$$\boldsymbol{v}' = \boldsymbol{P}_{a_5} \boldsymbol{P}_{a_1} \boldsymbol{P}_{a_3} \boldsymbol{v}$$

note

PFSMs $\Leftrightarrow$ *non-stationary / inhomogeneous discrete time Markov chains*

given this insight, it's time to switch gears . . .

random variables

random variable

$\Leftrightarrow$ a variable X whose value is subject to chance

$\Leftrightarrow$ a variable X that can assume various values x in a *sample space* $\mathcal{X}$, each according to some *probability* $p(X = x)$

disclaimer: yes, we write $\mathcal{X}$ instead of Ω

examples

here are (game-related) rvs X and their sample spaces $\mathcal{X}$

$health$	$\{low, medium, high\}$
$points$	$\{0, 1, \ldots, 100\}$
$location$	$\mathbb{R}^2$

$\Rightarrow$ sample spaces may be *univariate* or *multivariate* and (crucially) *discrete* or *continuous*

recall

for discrete rvs, we have

$$p : \mathcal{X} \to [0, 1]$$

$$\sum_{x \in \mathcal{X}} p(x) = 1$$

for continuous rvs, we have

$$p : \mathcal{X} \to \mathbb{R}_+$$

$$\int_{\mathcal{X}} p(x)\, dx = 1$$

for now, we will focus on discrete settings

Bayesian networks

Bayesian networks / graphical models

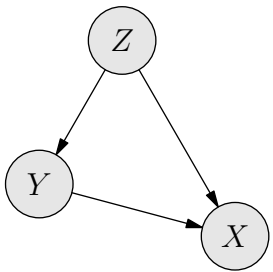
$\Leftrightarrow$ “conditional (in)dependence diagrams” for rvs

example

three rvs X, Y, Z whose *joint probability* can be modeled like this

$$p(X, Y, Z) = p(X \mid Y, Z)\, p(Y, Z) = p(X \mid Y, Z)\, p(Y \mid Z)\, p(Z)$$

$\Leftrightarrow$ *directed acyclic graph (DAG)* G



X is conditioned on Y and Z
Y is conditioned on Z

question

but what can we do with Bayesian networks ?

answer

many things . . . in particular, causal inference

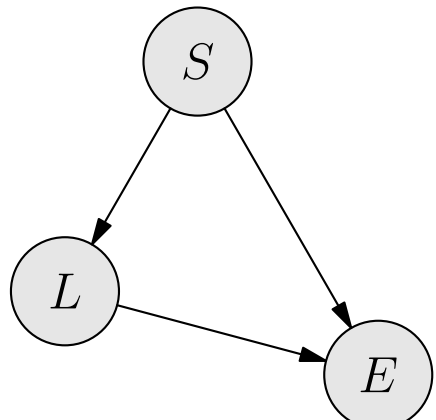
example (outdated in the age of strong AI)

what *causes* success in school ?

did you pass the last exam ($E \in \{t, f\}$) ?

did you learn for the exam ($L \in \{t, f\}$) ?

are you smart ($S \in \{t, f\}$) ?



probability tables (according to experience)

$p(S)$

S	
t	f
0.25	0.75

$p(L \mid S)$

S	L	
	t	f
t	0.70	0.30
f	0.60	0.40

$p(E \mid L, S)$

L	S	E	
		t	f
t	t	0.90	0.10
f	t	0.55	0.45
t	f	0.70	0.30
f	f	0.10	0.90

given that you passed, how likely are you smart ?

$$p\big(S = t \;\big|\; E = t\big)$$

given that you passed, how likely are you smart ?

$$p\big(S = t \;\big|\; E = t\big) = \frac{p(S = t, E = t)}{p(E = t)} \qquad \text{(product rule)}$$

given that you passed, how likely are you smart ?

$$p\big(S = t \;\big|\; E = t\big) = \frac{p(S = t, E = t)}{p(E = t)} \qquad \text{(product rule)}$$

$$= \frac{\sum_{L \in \{t,f\}} p(S = t, L, E = t)}{p(E = t)} \qquad \text{(marginalization)}$$

given that you passed, how likely are you smart ?

$$p\big(S = t \;\big|\; E = t\big) = \frac{p(S = t, E = t)}{p(E = t)} \qquad \text{(product rule)}$$

$$= \frac{\sum_{L \in \{t,f\}} p(S = t, L, E = t)}{p(E = t)} \qquad \text{(marginalization)}$$

$$= \frac{\sum_{L \in \{t,f\}} p(E = t|S = t, L)\, p(L|S = t)\, p(S = t)}{\sum_{L,S \in \{t,f\}} p(E = t|L, S)\, p(L|S)\, p(S)} \qquad \text{(marginalization)}$$

let's compute

$$p(S = t \mid E = t) = \frac{0.9 \cdot 0.7 \cdot 0.25 + 0.55 \cdot 0.3 \cdot 0.25}{0.9 \cdot 0.7 \cdot 0.25 + 0.55 \cdot 0.3 \cdot 0.25 + 0.7 \cdot 0.6 \cdot 0.75 + 0.1 \cdot 0.4 \cdot 0.75}$$

let's compute

$$p(S = t \mid E = t) = \frac{0.9 \cdot 0.7 \cdot 0.25 + 0.55 \cdot 0.3 \cdot 0.25}{0.9 \cdot 0.7 \cdot 0.25 + 0.55 \cdot 0.3 \cdot 0.25 + 0.7 \cdot 0.6 \cdot 0.75 + 0.1 \cdot 0.4 \cdot 0.75}$$

$$= \frac{0.1575 + 0.04125}{0.1575 + 0.04125 + 0.315 + 0.03}$$

$$= \frac{0.19875}{0.54375}$$

$$= 0.3655$$

given that you passed, how likely did you learn ?

$$p\big(L = t \;\big|\; E = t\big)$$

given that you passed, how likely did you learn ?

$$p\big(L = t \;\big|\; E = t\big) = \frac{p(L = t, E = t)}{p(E = t)}$$

given that you passed, how likely did you learn ?

$$p\big(L = t \;\big|\; E = t\big) = \frac{p(L = t, E = t)}{p(E = t)}$$

$$= \frac{\sum_{S \in \{t,f\}} p(S, L = t, E = t)}{p(E = t)}$$

$$= \frac{\sum_{S \in \{t,f\}} p(E = t|S, L = t)\, p(L = t|S)\, p(S)}{\sum_{S,L \in \{t,f\}} p(E = t|L, S)\, p(L|S)\, p(S)}$$

let's compute

$$
\begin{aligned}
p\big(L = t \,\big|\, E = t\big) &= \frac{0.9 \cdot 0.7 \cdot 0.25 + 0.7 \cdot 0.6 \cdot 0.75}{0.54375} \\
&= \frac{0.1575 + 0.315}{0.54375} \\
&= \frac{0.4725}{0.54375} \\
&\approx 0.869
\end{aligned}
$$

assignment

answer these questions

given that you are smart, how likely will you pass ?

given that you learned, how likely will you pass ?

if we map outcomes to numbers, e.g. $\{t, f\} \to \{1, 0\}$, we can also ask ...

$p(S)$

S	
1	0
0.25	0.75

$p(L \mid S)$

S	L	
	1	0
1	0.70	0.30
0	0.60	0.40

$p(E \mid L, S)$

L	S	E	
		1	0
1	1	0.90	0.10
0	1	0.55	0.45
1	0	0.70	0.30
0	0	0.10	0.90

what is the *expected value* for L given that you are smart and passed ?

$$\mathbb{E}\big[L \;\big|\; S = 1, E = 1\big]$$

what is the *expected value* for L given that you are smart and passed ?

$$\begin{aligned}
\mathbb{E}\big[L \;\big|\; S=1, E=1\big] &= \sum_{L\in\{0,1\}} L \cdot p\big(L \;\big|\; S=1, E=1\big) \\
&= 0 \cdot p\big(L=0 \;\big|\; S=1, E=1\big) + 1 \cdot p(L=1|S=1, E=1) \\
&= p\big(L=1 \;\big|\; S=1, E=1\big) \\
&= \cdots \\
&\approx 0.792
\end{aligned}$$

we have

$$p(L|S,E) = \frac{p(L,S,E)}{p(S,E)} = \frac{p(E|L,S)\,p(L,S)}{p(S,E)} = \frac{p(E|L,S)\,p(L|S)\,p(S)}{p(S,E)}$$

$$= \frac{p(E|L,S)\,p(L|S)\,p(S)}{\sum_L p(L,S,E)} = \frac{p(E|L,S)\,p(L|S)\,p(S)}{\sum_L p(E|L,S)\,p(L|S)\,p(S)}$$

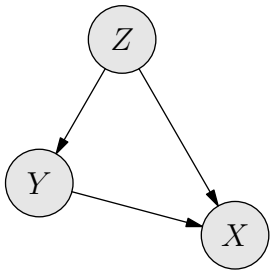
which now only involves quantities we can read from the given probability tables

once again

three rvs X, Y, Z with joint probability and factorization

$$p(X, Y, Z) = p(X \mid Y, Z)\, p(Y, Z) = p(X \mid Y, Z)\, p(Y \mid Z)\, p(Z)$$

$\Leftrightarrow$ *directed acyclic graph (DAG)* G



X is conditioned on Y and Z
Y is conditioned on Z

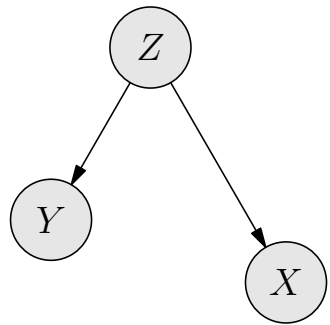
question

what if $(X \perp\!\!\!\perp Y) \mid Z$?

answer

we then simply have

$$\begin{aligned} p(X, Y, Z) &= p(X \mid Y, Z)\, p(Y, Z) \\ &= p(X \mid Z)\, p(Y \mid Z)\, p(Z) \end{aligned}$$

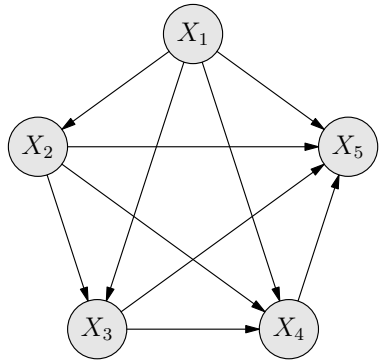


another example

five rvs $X_1, \ldots, X_5$ with joint probability and factorization

$$p(X_1, \ldots, X_5) = p(X_1)\, p(X_2|X_1)\, p(X_3|X_1, X_2)\, p(X_4|X_1, X_2, X_3)\, p(X_5|X_1, X_2, X_3, X_4)$$

$\Leftrightarrow$ DAG G



note

a DAG does not have directed cycles

$\Rightarrow$ we may *order* the vertices of a DAG

$\Leftrightarrow$ we may choose vertex identifiers

$$1, \ldots, n$$

such that

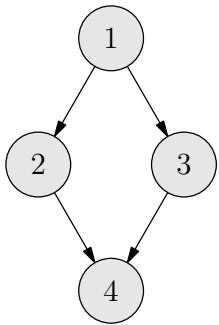
$$v_i \to v_j \;\Rightarrow\; i < j$$

this then yields an ordered DAG

in the following, we let

$$Pred(i) = \textit{set of predecessors of vertex } i$$

for example



$\Leftrightarrow$

$$Pred(4) = \{2, 3\}$$
$$Pred(3) = \{1\}$$
$$Pred(2) = \{1\}$$
$$Pred(1) = \varnothing$$

given a set of rvs $\mathcal{V} = \{X_1, X_2, \ldots, X_n\}$ and an index set $\mathcal{I} \subseteq \{1, 2, , \ldots, n\}$, we define

$$\mathcal{V}_{\mathcal{I}} \equiv \{X_i \in \mathcal{V} \mid i \in \mathcal{I}\}$$

for example

$$\mathcal{I} = \{2, 3\} \quad \Rightarrow \quad \mathcal{V}_{\mathcal{I}} = \{X_2, X_3\}$$
$$\mathcal{I} = Pred(4) \quad \Rightarrow \quad \mathcal{V}_{\mathcal{I}} = \{X_2, X_3\}$$

note

given random variables

$$\mathcal{V} = \{X_1, \ldots, X_n\}$$

with joint probability

$$p(X_1, \ldots, X_n)$$

and an ordered DAG G with n vertices,

we say $\mathcal{V}$ respects G or p respects G, if

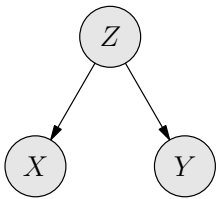
$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid \mathcal{V}_{Pred(i)})$$

for the fun of it . . .

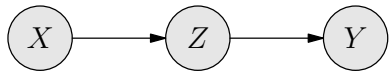
the following is not essential for this course,
but machine learners should know about it !

relations between 3 rvs X, Y, Z

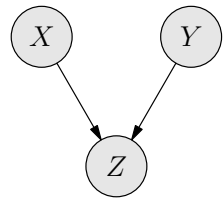
in the following DAGs, node Z is said to be of type



tail-tail



head-tail

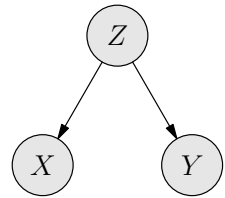


head-head

tail-tail

the DAG tells us

$$p(X, Y, Z) = p(Z)\, p(X \mid Z)\, p(Y \mid Z)$$



the prodcut rule tells us

$$p(X, Y, Z) = p(X, Y \mid Z)\, p(Z)$$

using both, we find

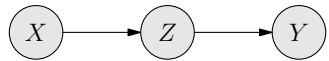
$$p(X, Y \mid Z) = p(X \mid Z)\, p(Y \mid Z)$$

$$\Rightarrow \quad (X \perp\!\!\!\perp Y) \mid Z \quad \Leftrightarrow \quad (Y \perp\!\!\!\perp X) \mid Z$$

head-tail

the DAG tells us

$$p(X, Y, Z) = p(X)\, p(Z \mid X)\, p(Y \mid Z)$$



the product rule tells us

$$p(X, Y, Z) = p(X, Y \mid Z)\, p(Z)$$

using both, we find

$$p(X, Y \mid Z) = \frac{p(X)\, p(Z|X)\, p(Y|Z)}{p(Z)} = \frac{p(X, Z)\, p(Y|Z)}{p(Z)} = \frac{p(X|Z)\, p(Z)\, p(Y|Z)}{p(Z)} = p(X|Z)\, p(Y|Z)$$

$$\Rightarrow \quad (X \perp\!\!\!\perp Y) \mid Z \quad \Leftrightarrow \quad (Y \perp\!\!\!\perp X) \mid Z$$

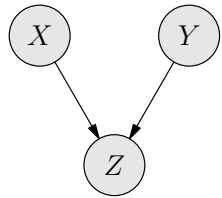
head-head

do we have

$$p(X, Y \mid Z) = p(X \mid Z)\, p(Y \mid Z) \quad ?$$

$\Leftrightarrow$ do we have

$$(X \perp\!\!\!\perp Y) \mid Z \quad ?$$



head-head

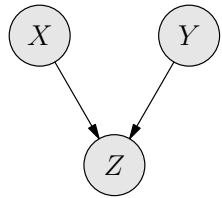
do we have

$$p(X, Y \mid Z) = p(X \mid Z)\, p(Y \mid Z) \quad ?$$

$\Leftrightarrow$ do we have

$$(X \perp\!\!\!\perp Y) \mid Z \quad ?$$

actually, not necessarily ...



example

toss two coins (behind a curtain)

$$X, Y \in \{0, 1\} \sim Bernoulli(0.5)$$

define a function of the outcome

$$Z = \texttt{XOR}(X, Y) = \begin{cases} 1 & \text{if } X \neq Y \\ 0 & \text{otherwise} \end{cases}$$

example

toss two coins (behind a curtain)

$$X, Y \in \{0, 1\} \sim \mathit{Bernoulli}(0.5)$$

define a function of the outcome

$$Z = \texttt{XOR}(X, Y) = \begin{cases} 1 & \text{if } X \neq Y \\ 0 & \text{otherwise} \end{cases}$$

now, assume we observe Z and then learn that $X = 1$

the two pieces of information immediately determine Y

$\Rightarrow X \not\perp\!\!\!\perp Y \mid Z$

summary

we now know about

difficulties associated with behavior modeling (and programming)

deterministic, non-deterministic, and probabilistic state machines

the notion of conditional (in)dependence of random variables

the notion of Bayesian networks or graphical models

the notion of a joint probability p *respecting* a DAG G