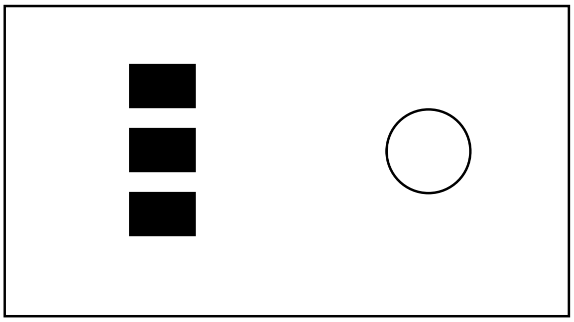
Learning a finite state device



Scenario: You have a box with buttons and lights. Box has internal state.

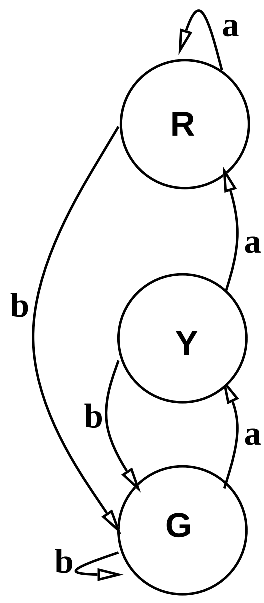
- $lights = f(current\ state)$
- $current\ state = g(button, prev\ state)$

E.g., Rubik's cube, robot in building, simple interactive agent.

Goal: build a predictive model of device (e.g., state diagram) by pressing buttons and observing lights.

An example w/o hidden state

2 actions: a, b.



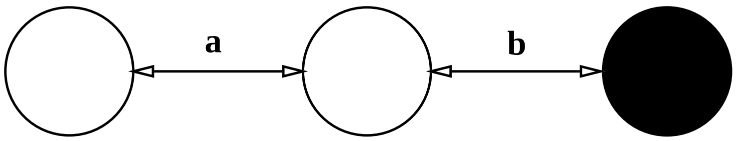
Generic algorithm when lights = state:

- Build a model.
- While not done: find an unexplored edge and take it.

What if several states look the same?

$$state \neq sensation$$

Another example



Some problems

What about redundancy? OK to learn logically equivalent state diagram.

What about hard-to-reach states?

One solution: Assume we can propose a hypothesis and receive a counterexample if it's not good enough.

(This is natural if we are going to use our learned model for some other task, or if we have test data we can use to test our hypotheses.)

Scenario

- Set of states Q. Start state q_0.
- Set of actions A.
- Transition function $\delta : Q \times A \to Q$.
- Observations: $obs(q)$, $obs(q, a_1 a_2 a_3 \ldots)$.

What does it mean for 2 states to be different?

q and q' are different if there is an action sequence a such that

$$obs(q, a) \neq obs(q', a).$$

Learn-DFA: Learning with a Reset

(assume can reset to q_0 whenever desired)

Idea: build DFA, where each state is "named" by:

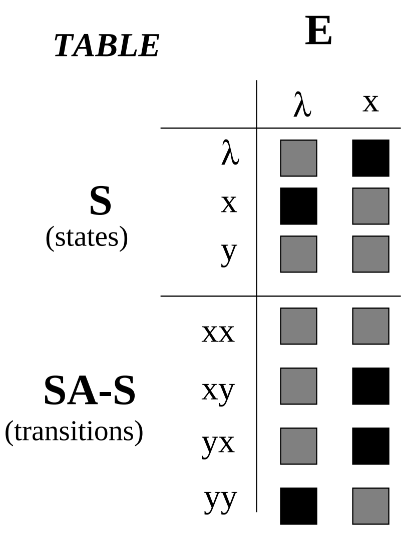
1. A string that gets you there.
2. The results of several experiments from that state.

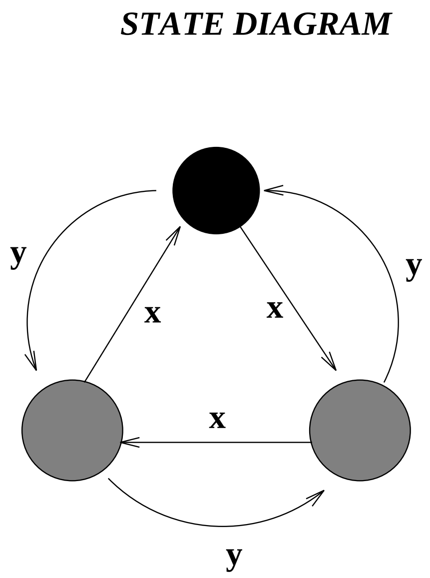
Use (2) to distinguish states that look the same.

Representation

- $S =$ set of strings a s.t. we *know* all states $\delta(q_0, a)$, for $a \in S$ are different.

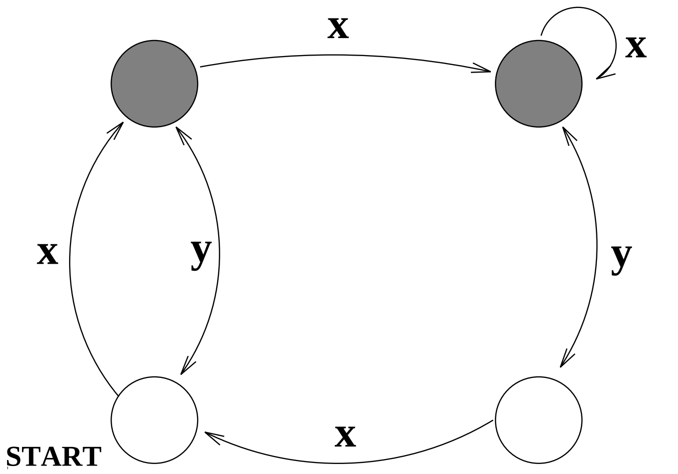
- $E =$ set of experiments used to distinguish them. (do S then E)





SA means strings $a_1 \ldots a_n$ where $a_1 \ldots a_{n-1} \in S$ and $a_n \in A$.

Example



Algorithm

Begin with $S = \{\lambda\}, E = \{\lambda\}$.

- while exists $s \in SA$ such that no $s' \in S$ has $row(s') = row(s)$, add s into S.
- Fill in transitions to make a hypothesis FSM.
- Query for counterexample z.
- Consider all splits of z into prefix p_i and suffix s_i, and replace p_i with its predicted equivalent $\alpha_i \in S$.
- Find $\alpha_i r_i$ and $\alpha_{i+1} r_{i+1}$ that produce different observations.
- Add r_{i+1} as a new experiment into E.

Getting rid of the reset:

- **Idea:** use "homing sequence": sequence of actions where observations uniquely determine ending state.

- Can learn homing sequence as you go.

Extensions: Other representations (e.g., Rubik's cube).

Related problems

- Noise / probabilistic transitions?
- Passive learning only? (e.g., HMMs)
- Goal is a policy rather than a model?
- What if state space is too large to keep explicitly?