Theory and Practice of Artificial Intelligence

Probabilistic Inference

Daniel Polani

School of Computer Science
University of Hertfordshire

March 9, 2017

Simplified Minesweeper

Example

1. Assume a mine is located at one of positions $1 \ldots 5$.
2. Its presence at position m is detectable at positions $m+1$ and $m-1$ (if in the field).
3. **Task:** Solve the minesweeper problem — find the mine.

Simplified Minesweeper: Bayes Model

Approach

1. $p(m)$ is the probability that the mine is at location m.
2. If we do not know anything, by symmetry: $p(m) = 1/5$.
3. A detection d can assume the values $\{\texttt{boom!}, \texttt{detect}, \texttt{nothing}\}$.
4. At position i, assuming the mine is at m, we have detection probability $p(d|m,i)$ with:

$$p(d|m,i) := \begin{cases} 1 & \text{if } d = f(m,i) \\ 0 & else \end{cases}$$

where

$$f(m,i) = \begin{cases} \texttt{boom!} & \text{if } m = i \\ \texttt{detect} & \text{if } |i-m| = 1 \\ \texttt{nothing} & \text{else}. \end{cases}$$

Simplified Minesweeper: $p(d|m,i)$

`boom!`

m \ i	1	2	3	4	5
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

`detect`

m \ i	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	0	0
3	0	1	0	1	0
4	0	0	1	0	1
5	0	0	0	1	0

`nothing`

m \ i	1	2	3	4	5
1	0	0	1	1	1
2	0	0	0	1	1
3	1	0	0	0	1
4	1	1	0	0	0
5	1	1	1	0	0

Simplified Minesweeper: Bayes' Theorem

1. Assume: uncovering position $i = 2$, we observe $d = \texttt{nothing}$.

2. Note that for each m, because of Bayes:

$$p(m|d, i = 2) = \frac{1}{Z} p(d|m, i = 2)p(m)$$

$$= Z^{-1} \left[p(d| \underset{\|}{m}_{1}, i) \cdot \frac{1}{5}, p(d| \underset{\|}{m}_{2}, i) \cdot \frac{1}{5}, p(d| \underset{\|}{m}_{3}, i) \cdot \frac{1}{5}, p(d| \underset{\|}{m}_{4}, i) \cdot \frac{1}{5}, p(d| \underset{\|}{m}_{5}, i) \cdot \frac{1}{5} \right]_{i=2}$$

3. Reminder

$$p(d|m, i) := \begin{cases} 1 & \text{if } d = f(m, i) \\ 0 & \text{else,} \end{cases} \qquad f(m, i) = \begin{cases} \texttt{boom!} & \text{if } m = i \\ \texttt{detect} & \text{if } |i - m| = 1 \\ \texttt{nothing} & \text{else.} \end{cases}$$

4.

$$\begin{aligned} p(m|d, i = 2) &= Z^{-1} \cdot [0, 0, 0, 0.2, 0.2] \\ &= [0, 0, 0, 0.5, 0.5] \end{aligned} \tag{1}$$

Simplified Minesweeper: Bayes' Theorem

1. Assume: uncovering position $i = 2$, we observe $d = \texttt{detect}$.
2. For each m (Bayes):

$$p(m|d, i = 2) = \frac{1}{Z} p(d|m, i = 2)p(m)$$
$$= Z^{-1} \left[p(d|m, i = 2)\big|_{m=1} \cdot \frac{1}{5}, p(d|m, i = 2)\big|_{m=2} \cdot \frac{1}{5}, p(d|m, i = 2)\big|_{m=3} \cdot \frac{1}{5}, \right.$$
$$\left. p(d|m, i = 2)\big|_{m=4} \cdot \frac{1}{5}, p(d|m, i = 2)\big|_{m=5} \cdot \frac{1}{5} \right]$$

3. Reminder

$$p(d|m, i) := \begin{cases} 1 & \text{if } d = f(m, i) \\ 0 & \text{else} \end{cases} \qquad f(m, i) = \begin{cases} \texttt{boom!} & \text{if } m = i \\ \texttt{detect} & \text{if } |i - m| = 1 \\ \texttt{nothing} & \text{else}. \end{cases}$$

4.

$$p(m|d, i = 2) = Z^{-1} \cdot [0.2, 0, 0.2, 0, 0] \\ = \quad [0.5, 0, 0.5, 0, 0] \qquad (2)$$

Simplified Minesweeper: Bayes' Theorem

1. Assume : uncovering position $i = 1$, we observe $d = \texttt{detect}$.
2. For each m (Bayes):

$$p(m|d, i = 1) = \frac{1}{Z} p(d|m, i = 1)p(m)$$
$$= Z^{-1} \left[p(d|m, i = 1)\big|_{m=1} \cdot \frac{1}{5}, p(d|m, i = 1)\big|_{m=2} \cdot \frac{1}{5}, p(d|m, i = 1)\big|_{m=3} \cdot \frac{1}{5}, \right.$$
$$\left. p(d|m, i = 1)\big|_{m=4} \cdot \frac{1}{5}, p(d|m, i = 1)\big|_{m=5} \cdot \frac{1}{5} \right]$$

3. Reminder

$$p(d|m, i) := \begin{cases} 1 & \text{if } d = f(m, i) \\ 0 & \text{else} \end{cases} \qquad f(m, i) = \begin{cases} \texttt{boom!} & \text{if } m = i \\ \texttt{detect} & \text{if } |i - m| = 1 \\ \texttt{nothing} & \text{else}. \end{cases}$$

4.

$$\begin{aligned} p(m|d, i = 1) &= Z^{-1} \cdot [0, 0.2, 0, 0, 0] \\ &= [0, 1, 0, 0, 0] \end{aligned} \tag{3}$$

Homework

Wumpus World

Check (Russell and Norvig 2002) **Wumpus World Revisited**, probabilistic treatment of the wumpus world.

Monty Hall Problem

Dude, Where Is My Car? (Monty Hall Problem)

Blackboard/Practical

Monty Hall (Python) I

```
from random import *
from rational import Rational

# in-line operation - rare use of side effect

def normalize(prob):
    Z = float(sum(prob[outcome]
                  for outcome in prob))
    for outcome in prob:
        prob[outcome] /= Z

# main

p = {}

p["cso"] = {}          # a joint probability for c,s,o, in
    this order
```

Monty Hall (Python) II

```
for run in xrange(1000000):
    doors = set(range(1,4))
    car = sample(doors,1)[0]          # sample and pick
    select = sample(doors,1)[0]

    open_door = sample(doors - set([car, select]),
        1).pop()

    outcome = (car, select, open_door)
    if not outcome in p["cso"]:
        p["cso"][outcome] = 0
    p["cso"][outcome] += 1

# normalize

normalize(p["cso"])
```

Monty Hall (Python) III

```
# seek: probability of where car is, conditioned on
    observed moves

# for this, we need p(s,o)

p["so"] = {}

for car, select, open_door in p["cso"]:
    observation = select, open_door
    if not observation in p["so"]:
        p["so"][observation] = 0
    p["so"][observation] += p["cso"][car, select,
        open_door]

normalize(p["so"])

# conditional
```

Monty Hall (Python) IV

```
p["c|so"] = {}

for car, select, open_door in p["cso"]:
    outcome = car, select, open_door
    p["c|so"][outcome] = p["cso"][outcome] /
        p["so"][select, open_door]

for car, select, open_door in p["c|so"]:
    if (select, open_door) == (1, 2):
        print car, p["c|so"][car, select, open_door]
```

Alternative Probability Classes (sampler.py)

```
class Sampler:
    def __init__(self):
        self.n = {}

    def __getitem__(self, x):
        return self.n[x]

    def observe(self, x):
        n = self.n
        if not x in n: n[x] = 0
        n[x] += 1
```

Alternative Probability Classes (prob2.py)

```
from sampler import *

class Prob:
    def __init__(self, sampler = None):
        if not sampler:
            self.p = {}
        else:
            self.p = dict((x, sampler[x]) for x in
                sampler.n)
        self.normalized = False

    def __getitem__(self, x):
        self.normalize()
        return self.p[x]

    def __setitem__(self, x, p):
        self.p[x] = p
        self.normalized = False
```

Usage of Alternative Probability Classes

```
from prob2 import *
import random

sampler = Sampler()

for i in xrange(100000):
    die = random.randint(1,6)
    sampler.observe(die)

p = Prob(sampler)


print p
```

Example: Markovian Self-Localization

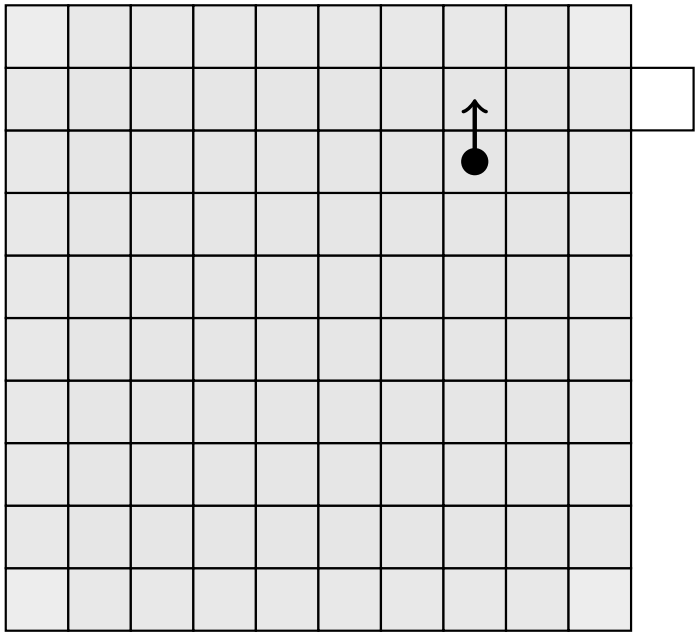
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

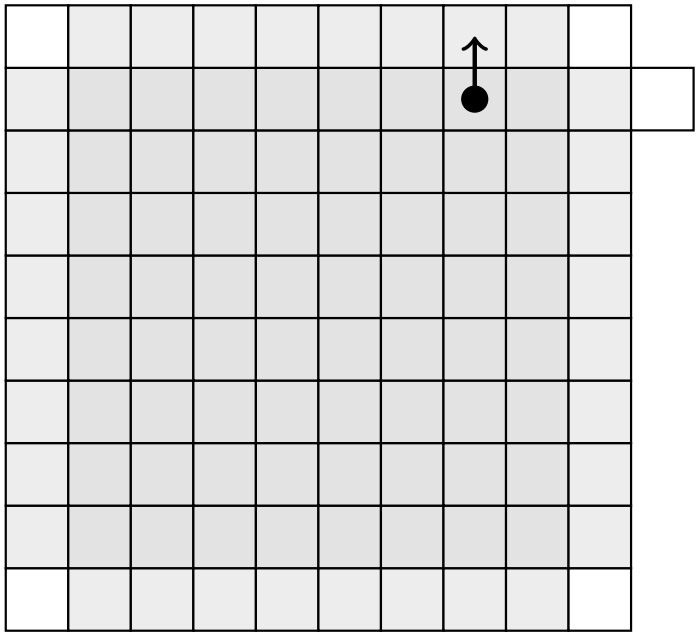
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

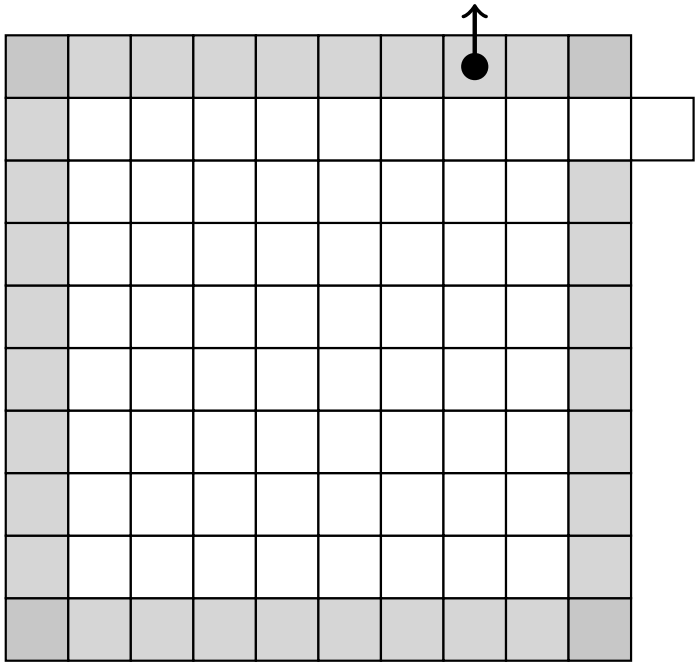
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

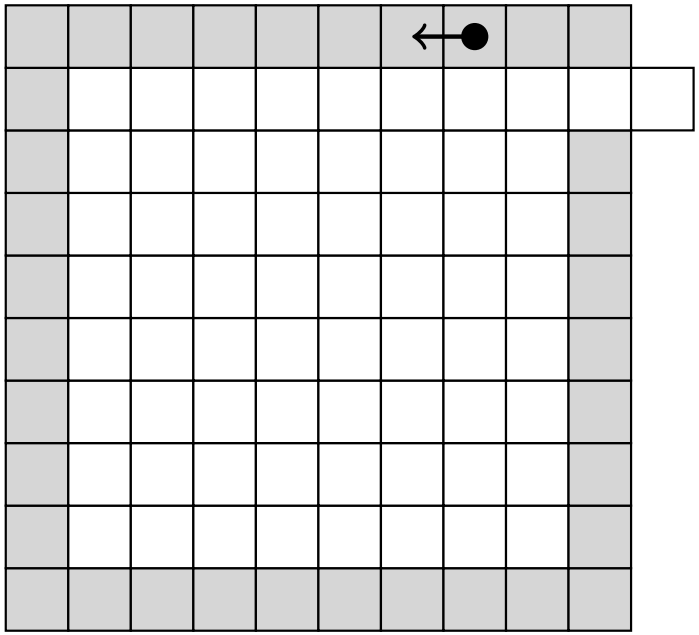
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

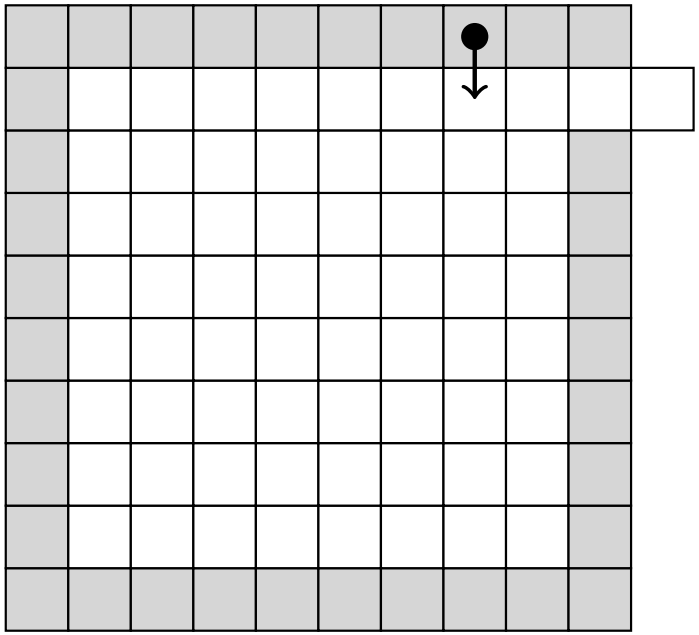
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

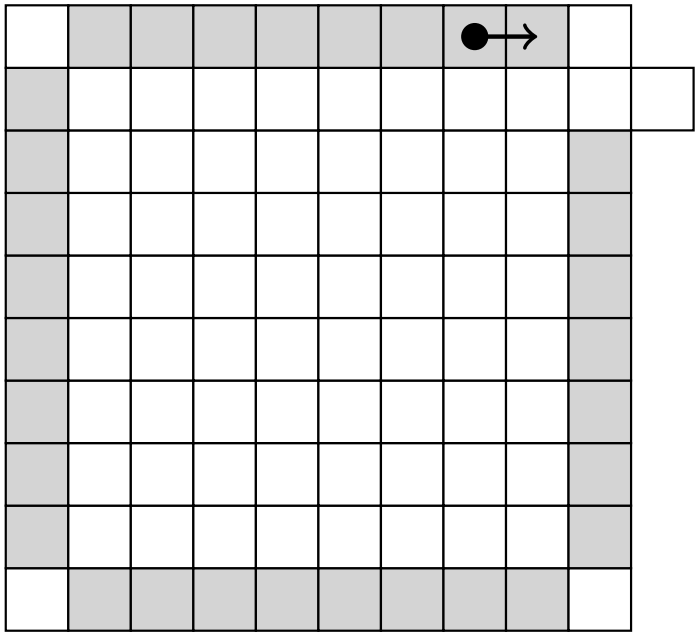
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

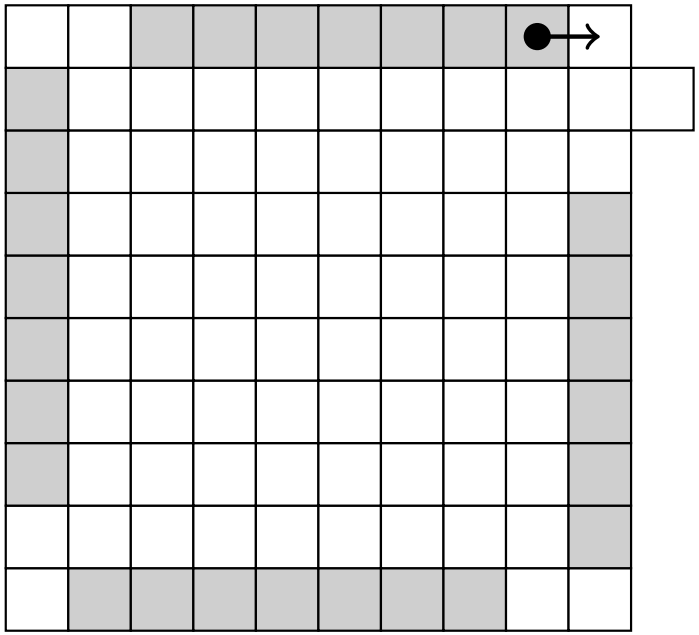
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

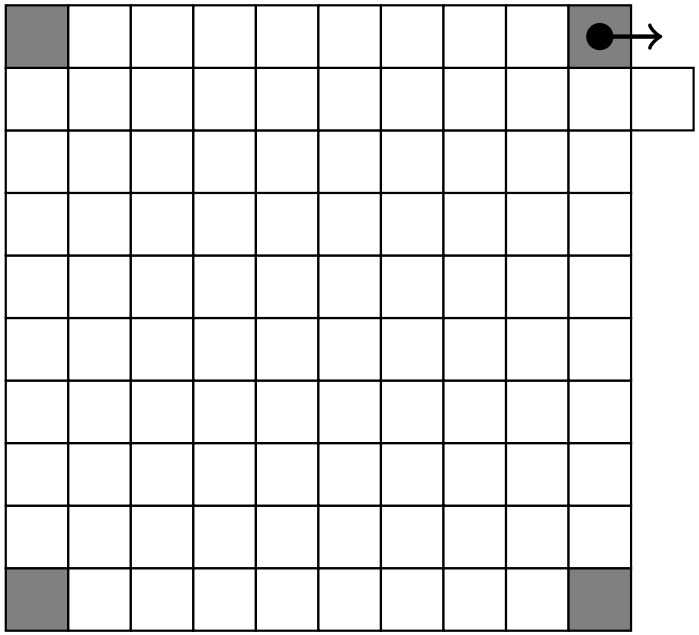
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

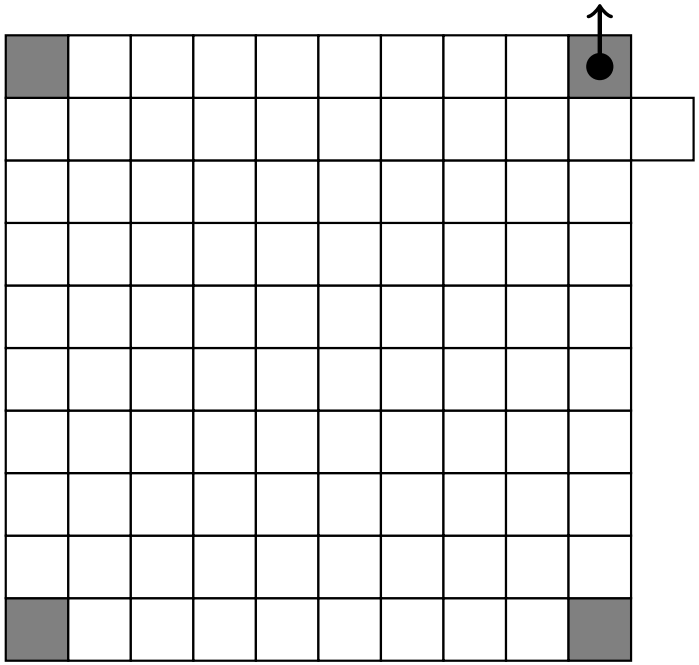
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

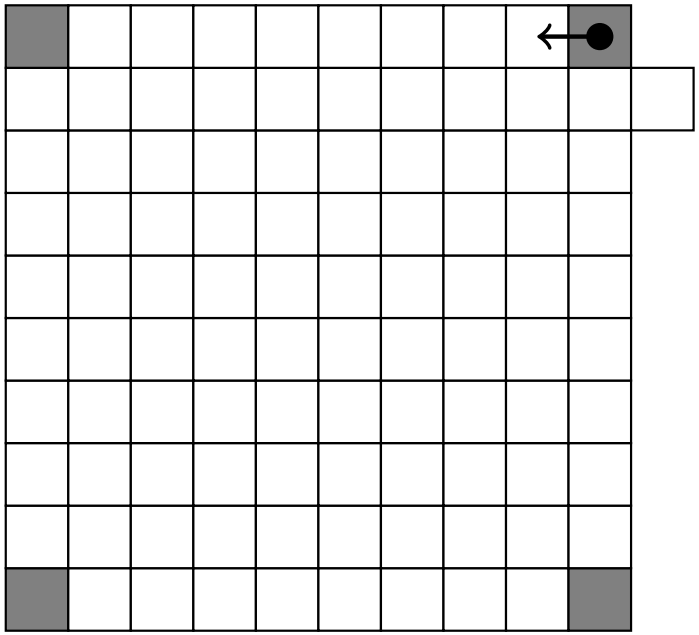
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

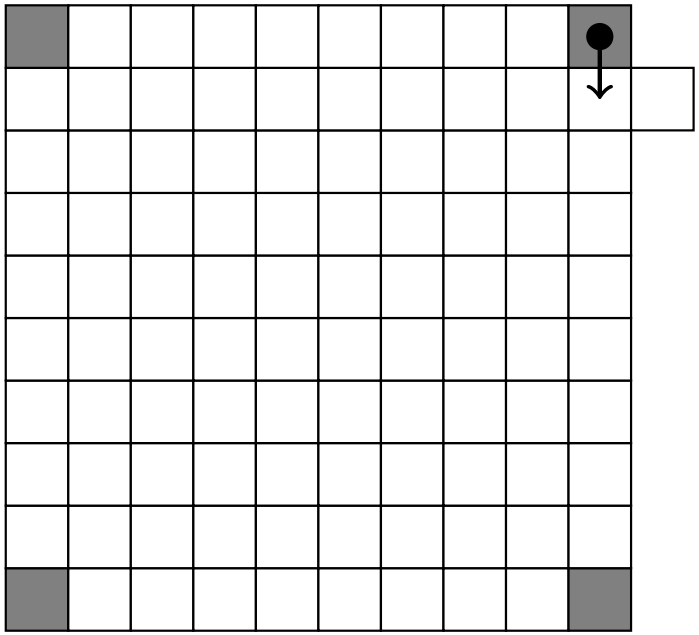
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

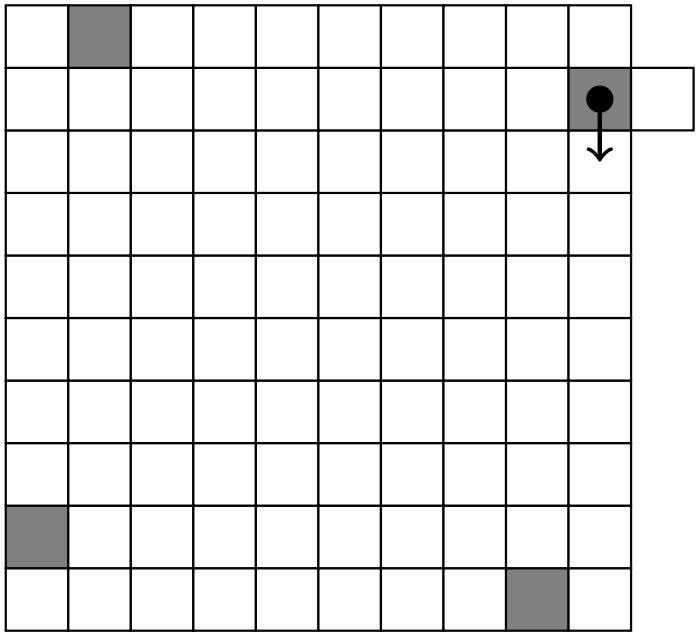
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Example: Markovian Self-Localization

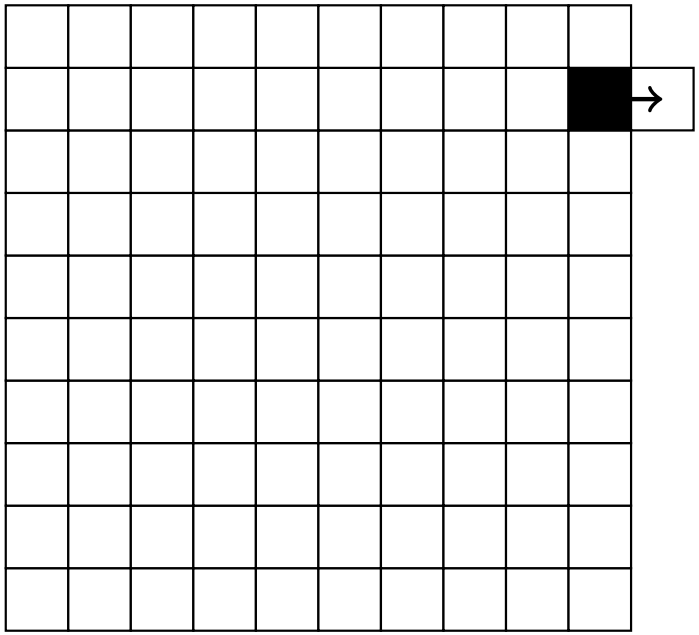
(Thrun et al. 2005)

Robot

- can move
- can turn
- can only detect `empty`, `wall`, `door` in front

Localization

- has knowledge about the map, but
- unknown position
- unknown orientation



Procedure

Preparation

1. consider set of states: $\mathcal{S} = \mathcal{R} \times \mathcal{D}$ with
   - location r
   - direction (orientation) d
   - together: state $s = (r, d)$
2. consider possible observations $o \in \mathcal{O} = \{\texttt{empty}, \texttt{wall}, \texttt{door}\}$
3. consider possible actions $a \in \mathcal{A} = \{\texttt{move}, \texttt{turn}\}$
4. define probability $p(o|s)$ that an observation o will be made if the state is s
5. define probability $p(s'|s,a)$ that the new state (location, orientation) of the robot will be s' if old state was s and action a is taken.
   more specifically: if action a is deterministic with $a(s) = s'$, then set

$$p(s'|s,a) := \begin{cases} 1 & \text{if } a(s) = s' \\ 0 & \text{else .} \end{cases}$$

Procedure II

Algorithm

1. start with probability prior $p(s)$ as the probability that the robot is at state (location, orientation) s
2. with an observation o, update the robot location probability with Bayes:

$$p(s|o) := \frac{1}{Z} p(o|s)p(s)$$

   where Z guarantees normalization, i.e. that $\sum_s p(s|o) = 1$. The posterior $p(s|o)$ becomes the new state probability of the robot after observation; call this $p(\tilde{s})$
3. update for the move (action) of the robot. The probability for the new state of the robot after performing action a is

$$p(s'|a) := \sum_{\tilde{s}} p(s'|\tilde{s}, a)p(\tilde{s})$$

   After choice of a, this distribution of states is our new model of what state the robot is in.
4. Use this as the new prior $p(s)$ and repeat loop from step 1

Notes

- steps 1 and 2 are the usual Bayesian update
- step 3 is additionally required to deal with movement of the robot

Part II

References

References I

Auer, P., (2003). Using Confidence Bounds for Exploitation-exploration Trade-offs. *J. Mach. Learn. Res.*, 3:397–422.

`http://dl.acm.org/citation.cfm?id=944919.944941`, Feb 2017

Bradberry, J., (2015). Introduction to Monte Carlo Tree Search.

`Introduction to Monte Carlo Tree Search`, 8. Feb. 2017

Browne, C., (2012). Monte Carlo Tree Search.

`https://pdfs.semanticscholar.org/0c6b/1e96cb05a266b410d25547dbf6bbc26ff0eb.pdf`, 8. Feb. 2017

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S., (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1).

References II

Kocsis, L., and Szepesvári, C., (2006). Bandit Based Monte-carlo Planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, 282–293. Berlin, Heidelberg: Springer-Verlag.

`http://dx.doi.org/10.1007/11871842_29`, Feb 2017

Pearl, J., (1984). *Heuristics: Intelligent Search Strategies for Computer Problem-Solving*. Addison Wesley.

Russell, S., and Norvig, P., (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall. Second edition.

Saygin, A., Cicekli, I., and Akman, V., (2000). Turing Test: 50 Years Later. *Minds and Machines*, 10(4):463–518.

Sedgewick, R., and Wayne, K., (2011). *Algorithms, 4th Edition*. Addison-Wesley.

References III

Sutton, R. S., and Barto, A. G., (1998). *Reinforcement Learning*. Cambridge, Mass.: MIT Press.

Thrun, S., Burgard, W., and Fox, D., (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press.