

Constructive Artificial Intelligence

Probabilistic Inference

Daniel Polani

School of Computer Science
University of Hertfordshire

November 23, 2009

All rights reserved. Permission is granted to copy and distribute these slides in full or in part for purposes of research, education as well as private use, provided that author, affiliation and this notice is retained.

Use as part of home- and coursework is only allowed with express permission by the responsible tutor and, in this case, is to be appropriately referenced.

Example

- ① Assume a mine is located at one of positions $1 \dots 5$.
- 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.

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 For a detection d_i at position i which can assume the values $\{\text{boom!}, \text{detect}, \text{nothing}\}$, we have:

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

where

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

Simplified Minesweeper III

Uncovering

- 1 Assume we decide to uncover position $i = 2$, we have observed d_2 which is nothing.
- 2 Note that for each m , because of Bayes:

$$\begin{aligned} p(m|d_2) &= \frac{1}{Z} p(d_2|m)p(m) \\ &= Z^{-1} \left[p(d_2|m)|_{m=1} \cdot \frac{1}{5}, p(d_2|m)|_{m=2} \cdot \frac{1}{5}, p(d_2|m)|_{m=3} \cdot \frac{1}{5}, p(d_2|m)|_{m=4} \cdot \frac{1}{5}, p(d_2|m)|_{m=5} \cdot \frac{1}{5} \right] \end{aligned}$$

- 3 Reminder

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

4

$$\begin{aligned} p(m|d_2) &= Z^{-1} \cdot [0, 0, 0, 0.2, 0.2] \\ &= [0, 0, 0, 0.5, 0.5] \end{aligned} \quad (1)$$

Simplified Minesweeper IV

Uncovering

- 1 Assume we decide to uncover position $i = 2$, we have observed d_2 which is detect.
- 2 For each m (Bayes):

$$\begin{aligned} p(m|d_2) &= \frac{1}{Z} p(d_2|m)p(m) \\ &= Z^{-1} \left[p(d_2|m)|_{m=1} \cdot \frac{1}{5}, p(d_2|m)|_{m=2} \cdot \frac{1}{5}, p(d_2|m)|_{m=3} \cdot \frac{1}{5}, p(d_2|m)|_{m=4} \cdot \frac{1}{5}, p(d_2|m)|_{m=5} \cdot \frac{1}{5} \right] \end{aligned}$$

- 3 Reminder

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

4

$$\begin{aligned} p(m|d_2) &= Z^{-1} \cdot [0.2, 0, 0.2, 0, 0] \\ &= [0.5, 0, 0.5, 0, 0] \end{aligned} \tag{2}$$

Simplified Minesweeper V

Uncovering

- 1 Assume we decide to uncover position $i = 1$, we have observed d_2 which is detect.
- 2 For each m (Bayes):

$$\begin{aligned} p(m|d_2) &= \frac{1}{Z} p(d_2|m)p(m) \\ &= Z^{-1} \left[p(d_2|m)|_{m=1} \cdot \frac{1}{5}, p(d_2|m)|_{m=2} \cdot \frac{1}{5}, p(d_2|m)|_{m=3} \cdot \frac{1}{5}, p(d_2|m)|_{m=4} \cdot \frac{1}{5}, p(d_2|m)|_{m=5} \cdot \frac{1}{5} \right] \end{aligned}$$

- 3 Reminder

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

- 4

$$\begin{aligned} p(m|d_2) &= Z^{-1} \cdot [0, 0.2, 0, 0, 0] \\ &= [0, 1, 0, 0, 0] \end{aligned} \tag{3}$$

Wumpus World

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

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

Blackboard/Practical

Monty Hall (Python) I

```
import psyco
from random import *
from rational import Rational
psyco.full()

# 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

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
```

Monty Hall (Python) II

```
# normalize

normalize(p["cso"])

# 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] += 1

normalize(p["so"])

# conditional

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

    def __repr__(self):
        self.normalize()
        p = self.p
        name = tuple(self.__dict__)[0]
        return "\n".join(["%s(%s) = %f" % (name, str(x), p[x]) for x in p])

    def normalize(self):
        if self.normalized: return
        p = self.p
        Z = float(sum(p[x] for x in p))
        for x in p: p[x] /= Z
        self.normalized = True
```

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
```

References

- Cover, T. M., and Thomas, J. A., (1991). *Elements of Information Theory*. New York: Wiley.
- Jaynes, E. T., (1957). Information theory and statistical mechanics. *Phys. Rev.*, 106(4):620–630.
- 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.
- Shannon, C. E., (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423.