COMP14112

Artificial Intelligence Fundamentals

Lecture Notes - Part One

Second Semester
2013-2014

School of Computer Science
University of Manchester

COMP14112: Artificial Intelligence Fundamentals

Lecture 0 –Very Brief Overview

Lecturer: **Xiao-Jun Zeng**
Email: **x.zeng@manchester.ac.uk**

Overview

- This course will focus mainly on probabilistic methods in AI
  - We shall present probability theory as a general theory of reasoning
  - We shall develop (recapitulate) the mathematical details of that theory
  - We shall investigate two applications of that probability theory in AI which rely on these mathematical details:
    - Robot localization
    - Speech recognition

Structure of this course

- Part One -The first 4 weeks + week 10

Week	Lecture	Examples class	Laboratory exercise
1	Robot localization I		
2	Robot localization II	Probability I	
3	Foundations of probability	Probability II	1.1 Robot localization I
4	Brief history of AI	Probability III	
5	……	Turing's paper	1.2D Robot localization II
…..	……	…..	
10	Part 1 Revision	Revision	

COMP14112: Artificial Intelligence Fundamentals

Lecture 1 - Probabilistic Robot Localization I

Probabilistic Robot Localization I

Outline

- Background
- Introduction of probability
  - Definition of probability distribution
  - Properties of probability distribution
- Robot localization problem

Background

- Consider a mobile robot operating in a relatively static (hence, known) environment cluttered with obstacles.

Background

- The robot is equipped with (noisy) sensors and (unreliable) actuators, enabling it to perceive its environment and move around within it.
- We consider perhaps the most basic question in mobile robotics: how does the robot know where it is?
- This is a classic case of reasoning under uncertainty: almost none of the information the robot has about its position (its sensor reports and the actions it has tried to execute) yield certain information.
- The theory we shall use to manage this uncertainty is **probability theory**.

Introduction of Probability

- **Sample space**
  - **Definition**. The **sample space**, Ω , is the set of all possible outcomes of an experiment.
  - **Example**. Assume that the experiment is to check the exam marks of students in the school of CS, then the sample space is

$$\Omega = \{s \mid s \text{ } is \text{ } a \text{ } CS \text{ } student\}$$

In the following, let *M(s)* represents the exam mark for student *s* and a be a natural number in [0, 100], we define

$$\{a\} \equiv \{s \mid M(s) = a\}$$
$$\{\geq a\} \equiv \{s \mid M(s) \geq a\}$$
$$\{< a\} \equiv \{s \mid M(s) \geq a\}$$
$$\{[a, b]\} \equiv \{s \mid a \leq M(s) \leq b\}$$

Introduction of Probability

- **Event**:
  - **Definition**. An **event** , *E*, is any subset of the sample space Ω .
  - **Examples**
    - Event 1={40}, i.e., the set of all students with 40% mark
    - Event 2={≥40}, i.e., the set of all students who have passed

Definition of probability distribution

- **Probability distribution**
  - **Definition**. Given sample space Ω , a ***probability distribution*** is a function $p(E)$ which assigns a real number in [0,1] for each event E *in* Ω and satisfies

    (i) $P(\Omega) = 1$ (K1)

    (ii) If $E1 \wedge E2 = \phi$ (i.e., if $E1$ and $E2$ are mutually exclusive), then

$$p(E1 \vee E2) = p(E1) + p(E2) \qquad \text{(K2)}$$

    where $E1 \wedge E2$ means $E1 \textit{ and } E2$; $E1 \vee E2$ means $E1 \textit{ or } E2$.
  - **Example.** Let $p(E)$ as the percentage of students whose marks are in E , then $p(E)$ is a probability distribution on Ω .

Definition of Probability Distribution

- **Some notes on probability distribution**
  - For an event *E*, we refer to *p(E)* as the probability of *E*
  - Think of *p(E)* as representing one's **degree of belief** in *E*:
    - if *p(E) = 1*, then *E* is regarded as certainly true;
    - if *p(E) = 0.5*, then *E* is regarded as just as likely to be true as false;
    - if *p(E) = 0*, then *E* is regarded as certainly false.
  - So the probability can be experimental or subjective based dependent on the applications

Properties of Probability Distribution

- **Basic Properties of probability distribution:** Let $p(E)$ be the probability on Ω , then
  - For any event E, $p(E^C) = 1 - p(E)$, where E^C is complementary (i.e., not E) event;
  - If events $E \subseteq F$, then $p(E) \leq p(F)$;
  - For any two events E and F ,

$$p(E \vee F) = p(E) + p(F) - p(E \wedge F)$$

  - For empty event (i.e., empty set) ϕ , $p(\phi) = 0$

Properties of Probability Distribution

- **Example:** Let $\Omega = \{s \mid s \text{ is a CS student}\}$ be the sample space for checking the exam marks of students, Let:
  - $E = \{\geq 40\}$, *i.e., the event of "pass"*
  - $F = \{\geq 70\}$, *i.e., the event of "1st class"*
  - Then $p(F) \leq p(E)$ as $F \subseteq E$
  - If $p(E) = 0.75$ $p(F) = 0.10$
  - then the probability of event $G = \{< 40\}$ (i.e., the event of "fail") is

$$p(G) = p(E^C) = 1 - p(E) = 1 - 0.75 = 0.25$$

12

Properties of Probability Distribution

- **Example (continue) :** Assume

$E = \{\geq 40\}$ $F = \{\geq 70\}$

$p(E) = 0.75,$ $p(F) = 0.10$

  - *Question: what is the probability of event*

$H = \{< 40\} \vee \{\geq 70\}$ *i.e., the event of "fail or 1st class"*

  - *Answer:*

$$\text{As } E^C \wedge F = \{< 40\} \wedge \{\geq 70\} = \phi,$$
$$\text{Then the probability of event H is}$$
$$p(H) = p(E^C \vee F) = p(E^C) + p(F)$$
$$= 0.25 + 0.10 = 0.35$$

13

Properties of Probability Distribution

- The following notions help us to perform basic calculations with probabilities.
- **Definition**:
  - Events $E_1, E_2, ..., E_n$ are **mutually exclusive** if

$$p(E_i \wedge E_j) = 0 \quad for \quad i \neq j$$

  - Events $E_1, E_2, ..., E_n$ are **jointly exhaustive** if

$$p(E_1 \vee E_2 \vee ... \vee E_n) = 1$$

  - Events $E_1, E_2, ..., E_n$ form a **partition** (of Ω) if they are mutually exclusive and jointly exhaustive.
- **Example**. Events $E_i = \{i\}, i = 0, 1, ..., 100$ form a partition of Ω which is defined in the previous examples.

14

Properties of Probability Distribution

- **Property related to partition**:
  - If Events $E_1, E_2, ..., E_n$ are mutually exclusive, then

$$p(E_1 \vee E_2 \vee ... \vee E_n) = p(E_1) + p(E_2) + ... + p(E_n)$$

  - If events $E_1, E_2, ..., E_n$ form a partition

$$p(E_1) + p(E_2) + ... + p(E_n) = 1$$

15

Properties of Probability Distribution

- **Example**: Consider the following events
  - $E_1 = \{< 40\}$, *i.e., the event of "fail"*
  - $E_1 = \{[41,59]\}$, *i.e., the event of "pass but less than 2.1"*
  - $E_3 = \{[60,69]\}$, *i.e., the event of "2.1"*
  - $E_4 = \{[70,100]\}$, *i.e., the event of "1st class"*

  Then E_1, E_2, E_3, E_4 form a **partition** of Ω
  - If $p(E_1) = 0.25,$ $p(E_2) = 0.4,$ $p(E_3) = 0.25$, then

$$p(E_4) = 1 - [p(E_1) + p(E_2) + p(E_3)] = 0.10$$

  - Further let $E_5 = \{\geq 40\}$ (i.e., the event of "pass"), then

$$p(E_5) = p(E_2 \vee E_3 \vee E_4) = p(E_2) + p(E_3) + p(E_4) = 0.75$$

16

Robot Localization Problem

- The robot localization problem is one of the most basic problems in the design of autonomous robots:
  - Let a robot equipped with various sensors move freely within a known, static environment. Determine, on the basis of the robot's sensor readings and the actions it has performed, its current pose (position and orientation).
  - In other words: Where am I?

17

Robot Localization Problem

- Any such robot must be equipped with sensors to perceive its environment and actuators to change it, or move about within it.
- Some types of actuator
  - DC motors (attached to wheels)
  - Stepper motors (attached to gripper arms)
  - Hydraulic control
  - ‘Artificial muscles’
  - . . .

Robot Localization Problem

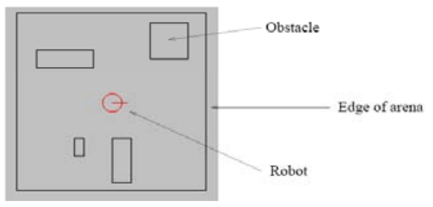
- Some types of sensor
  - Camera(s)
  - Tactile sensors
    - Bumpers
    - Whiskers
  - Range-finders
    - Infra-red
    - Sonar
    - Laser range-finders
  - Compasses
  - Light-detectors

Robot Localization Problem

- In performing localization, the robot has the following to work with:
  - knowledge about its initial situation
  - knowledge of what its sensors have told it
  - knowledge of what actions it has performed
- The knowledge about its initial situation may be incomplete (or inaccurate).
- The sensor readings are typically noisy.
- The effects of trying to perform actions are typically unpredictable.

Robot Localization Problem

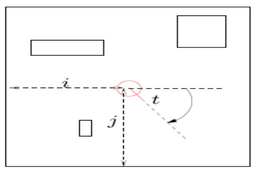
- We shall consider robot localization in a simplified setting: a single robot equipped with rangefinder sensors moving freely in a square arena cluttered with rectangular obstacles:



- In this exercise, we shall model the robot as a point object occupying some position in the arena not contained in one of the obstacles.

Robot Localization Problem

- We impose a 100×100 square grid on the arena, with the grid lines numbered 0 to 99 (starting at the near left-hand corner).
- We divide the circle into 100 units of π/50 radians each, again numbered 0 to 99 (measured clockwise from the positive x-axis.
- We take that the robot always to be located at one of these grid intersections, and to have one of these orientations.
- The robot’s pose can then be represented by a triple of integers *(i, j, t)* in the range [0, 99], thus: as indicated.



Robot Localization Problem

- Now let us apply the ideas of probability theory to this situation.
- Let $L_{i,j,t}$ be the event that the robot has pose *(i, j, t)* (where *i, j, t* are integers in the range [0,99]).
- The collection of events

$$\{L_{i,j,t} \mid 0 \leq i, j, t < 100\}$$

  forms a partition!
- The robot will represent its beliefs about its location as a probability distribution
- Thus, $p(L_{i,j,t})$ is the robot’s degree of belief that its current pose is *(i, j, t)*.

Robot Localization Problem

- The probabilities $p(L_{i,j,t})$ can be stored in a 100 × 100 × ×100-matrix.
- But this is hard to display visually, so we proceed as follows.
- Let $L_{i,j}$ be the event that the robot has position *(i, j),* and let L_t be the event that the robot has orientation *t*
- Thus

$$L_{i,j} \equiv \bigvee_t L_{i,j,t}$$

$$L_t \equiv \bigvee_i \bigvee_j L_{i,j,t}$$

24

Robot Localization Problem

- As the events $L_{i,j,t}$ form a partition, then, based on the property related to partition, we have

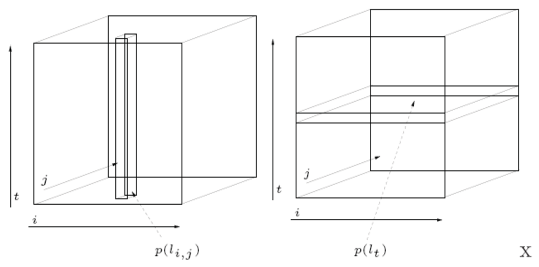
$$P(L_{i,j}) = P\left(\bigvee_t L_{i,j,t}\right) = \sum_t P(L_{i,j,t})$$

$$P(L_t) = P\left(\bigvee_i \bigvee_j L_{i,j,t}\right) = \sum_i \sum_j P(L_{i,j,t})$$

25

Robot Localization Problem

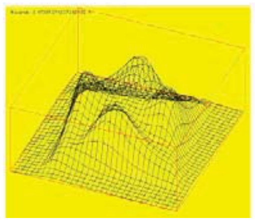
- These summations can be viewed graphically:

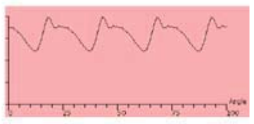


26

Robot Localization Problem

- The robot's degrees of belief concerning its position can then be viewed as a surface, and its degrees of belief concerning its orientation can be viewed as a curve, thus:



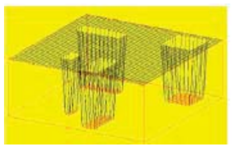


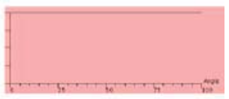
- The question before us is: how should the robot assign these degrees of belief?

27

Robot Localization Problem

- There are two specific problems:
  - what probabilities does the robot start with?
  - how should the robot's probabilities change?
- A reasonable answer to the first question is to assume all poses equally likely, except those which correspond to positions occupied by obstacles:





- We shall investigate the second question in the next lecture.

28

COMP14112: Artificial Intelligence Fundamentals

Lecture 2 - Probabilistic Robot Localization II

Lecturer: Xiao-Jun Zeng
Email: x.zeng@manchester.ac.uk

Probabilistic Robot Localization II

Outline

- Revisit the last lecture
- Introduction of conditional probability
  - Definition of conditional probability
  - Properties of conditional probability
  - Bayes' Theorem
- Introduction of random variables
  - Definition of random variables
  - Discrete random variables
  - Continuous random variables
- Robot localization problem

1

Last Lecture

At the end of the last lecture, it was pointed out

- There are two specific problems for robot localization:
  – what probabilities does the robot start with?
  – how should the robot's probabilities change?
- A reasonable answer to the first question is to assume all poses equally likely, except those which correspond to positions occupied by obstacles.
- To answer the second question, we need to have a method which can update the robot's probabilities, based on the condition of the current observations from the sensors. Such a method is the theory of **conditional probability**.

2

Definition of Conditional Probability

- **Conditional Probability** is the probability of some event F, given the occurrence of some other event E. *Conditional probability* is written P(F|E)**.**
- **Example**. An experiment consists of rolling a die once. Let X be the outcome. Assume
  – F be the event {X = 6} and E be the event {X > 4}.
  – The probability of each event {X=i} (i=1,2,…,6) is the same as 1/6. Then p(F)=1/6
  – the die is rolled and we are told that event E has occurred.

  This leaves only two possible outcomes: 5 and 6.
  – So under the occurrence of event E, the probability of F becomes 1/2, making P(F|E) = 1/2.

3

Definition of Conditional Probability

- **Definition**: Given a probability distribution p and two events E and F with $p(E) > 0$, the conditional probability of F given E is defined by

$$p(F \mid E) = \frac{p(F \wedge E)}{p(E)}$$

- **Example** (continue). Recall that F is the event {X = 6}, and E is the event {X >4}. Note that E ^ F=F. So, the above formula gives

$$p(F \mid E) = \frac{p(F \wedge E)}{p(E)} = \frac{p(F)}{p(E)} = \frac{1/6}{1/3} = \frac{1}{2}$$

in agreement with the calculations performed earlier.

4

Properties of Conditional Probability

- **Basic properties of conditional probability**: Let p a probability distribution, let E and F be events with $p(E) > 0$, then
  1. $0 \le p(F \mid E) \le 1$;
  2. If $p(F) = 0$, then $p(F \mid E) = 0$;
  3. If $E \subseteq F$, then $p(F \mid E) = 1$;

- Think of $p(F \mid E)$ as the degree of belief an agent *would* have in F *if and when* it learned that E occurred.

5

Properties of Conditional Probability

- We should check that this method of updating probability distributions really leaves with a probability distribution.
- **Property 1**: Let p be a probability distribution on Ω, and let E_0 be an event on Ω with $p(E_0) > 0$. Define the function

$$p_{E_0}(E) = p(E \mid E_0)$$

for all event E on Ω. Then p_{E_0} is a probability distribution. It is called the probability distribution obtained by conditionalizing on E_0.

6

Properties of Conditional Probability

- The foregoing discussion suggests the following policy of belief-revision:
  – If an agent's degrees of belief are given by the probability distribution p, and E_0 is an event representing the agent's total new information at some point in time, with $p(E_0) > 0$, then the agent's new degrees of belief should be given by the probability distribution p_{E_0} obtained by conditionalizing on E_0.
- This policy is sometimes known as Bayesian updating, after Thomas Bayes.

7

Properties of Conditional Probability

- Conditionalizing has many pleasant properties.
- **Property 2**: Let p be a probability distribution, and let E, F, and G be events with $p(F \wedge G) > 0$, then

$$(p_F)_G(E) = p_{F \wedge G}(E) = (p_G)_F(E)$$

- Thus, if an agent receives several new pieces of information, it does not matter which order it conditionalizes in.

8

Properties of Conditional Probability

- The following identities are fundamental.
- **Property 3** (Total Probability Formula):
  – Let p be a probability distribution and $E, E_1, ..., E_n$ be events. If $E_1, ..., E_n$ form a partition and $p(E_1), ..., p(E_n)$ are all positive. Then

$$p(E) = p(E \mid E_1)p(E_1) + ... + p(E \mid E_n)p(E_n)$$

  – Total Probability formula suggests that, if the partition $E_1, ..., E_n$ is well understood, then finding an unconditional probability can be done from the conditional ones.

9

Properties of Conditional Probability

- Similarly
- **Property 4** (Extended Total Probability Formula):
  – Let p be a probability distribution and $E, F, E_1, ..., E_n$ be events. If $E_1, ..., E_n$ form a partition and $p(F \wedge E_1), ..., p(F \wedge E_n)$ all positive. Then

$$p(E \mid F) = p(E \mid F \wedge E_1)p(E_1 \mid F) + ... + p(E \mid F \wedge E_n)p(E_n \mid F)$$

10

Bayes' Theorem

- The next theorem is probably the most important (yet simple!) in probability theory.
- **Bayes' Theorem**. Let p be a probability distribution, and let E and F be two events, with $p(E)$ and $p(F)$ both positive. Then

$$p(E \mid F) = \frac{p(F \mid E)p(E)}{p(F)}$$

11

Bayes’ Theorem

- Bayes’ Theorem has an alternative form, which is often more useful.
- **Bayes’ Theorem - alternative form**:
  - Let p be a probability distribution and $E, E_1,\dots, E_n$ be events. If $E_1,\dots, E_n$ form a partition and $p(E_1),\dots, p(E_n)$ are all positive. Then

$$p(E_i \mid E) = \frac{p(E \mid E_i)p(E_i)}{p(E \mid E_1)p(E_1) + \dots + p(E \mid E_n)p(E_n)}$$

12

Bayes’ Theorem

- Bayes’ theorem is typically employed when the $E_1,\dots, E_n$ form a partition of conflicting **hypotheses** and E represents an observation that will help us decide between these hypotheses.
- In many situations, it is clear what the conditional probabilities $p(E \mid E_i)$ should be.
- • But, of course, what we want to find are the quantities $p(E_i \mid E)$.
- • Bayes’ theorem allows us to do exactly this.

13

Bayes’ Theorem

- **Example**: A patient presents at a clinic exhibiting a symptom s , for which three possible diseases come into question: d_1 ,d_2 and d_3 . Suppose that we know these diseases cannot occur together, and suppose further that we have statistics telling us both how likely symptom s is given each disease, and also how widespread each disease is in the general population. Using s, d_1, d_2, d_3 to stand for the obvious events, we know:

$$p(s \mid d_1),\ p(s \mid d_2),\ p(s \mid d_3),\ p(d_1),\ p(d_2),\ p(d_3),$$

Bayes’ theorem then allows us to compute what we want:

$$p(d_i \mid s) = \frac{p(s \mid d_i)p(d_i)}{p(s \mid d_1)p(d_1) + p(s \mid d_2)p(d_2) + p(s \mid d_3)p(d_3)}$$

14

Definition of Random Variables

- **Definition**. A **random variable** (r.v) is a function $X(\cdot)$ which assigns to each element ω of sample space Ω a real value $X(\omega)$.
- **Definition**. For any random variable X , we define the **cumulative distribution function** (CDF) $F_X(x)$, as

$$F_X(x) \equiv p(X \le x) \equiv p(\{\omega \in \Omega : X(\omega) \le x\})$$

where $x \in R$ (i.e., the space of all real numbers).

- **Example.** Let the sample space Ω be all students in the School of CS. For each student ω , define a function $X(\cdot)$ whose value is the exam mark. Then

  X is a random variable and CDF $F_X(x)$ is the percentage of students’ marks less than x

15

Type of Random Variables

- **Types of Random Variables**: Discrete & Continuous
  - A **discrete** rv is one whose possible values constitute a finite set or a countable infinite set
  - A **continuous** rv is one whose possible values consists of an entire interval on the real line
- **Example**
  - The student mark X is a discrete rv.
  - Let X be the temperature of a given day at Manchester, then X is a continuous rv.

16

Discrete Random Variables

- **Definition**. The probability mass function (pmf) of a discrete rv X is given by

$$f_X(x_i) \equiv p(X = x_i)$$

- The properties of discrete rv are fully determined by its probability mass function.
- **Example**. Let X be a diecrete rv taking its values as 1,2,…,n. If its pmf is given by $f_X(i) = 1/n,\ i = 1,2,\dots, n$ then it is rv with uniform distribution.

17

Continuous Random Variables

- **Definition**. The probability density function (pdf) of a continuous random variable X is given by

$$f_X(x) \equiv \frac{d}{dx} F_X(x)$$

- Basic properties of pdf:

$$f_X(x) \geq 0, x \in R$$

$$\int_{-\infty}^{\infty} f_X(x)dx = 1$$

- The properties of a continuous rv are fully determined by its probability density function.

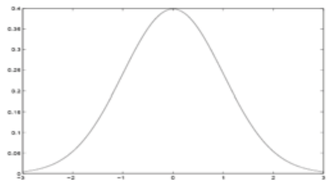
18

Continuous Random Variables

- **Example**. Let X be a continuous rv taking its values as real numbers. It is called as a rv with normally distributed if its probability density function is given by

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is a real number and σ is a positive number.



19

Mean and Variance

- The **mean** or **expectation** of rv X is defined as

$$\mu = E(X) = \begin{cases} \sum_i x_i f_X(x_i), & \text{discrete} \\ \int_{-\infty}^{\infty} x f_X(x)\, dx, & \textit{continuous} \end{cases}.$$

- The average spread of rv X around its expectation is called the **standard deviation**, and is computed via the so-called **variance**

$$\text{Variance} \quad \sigma^2 = V(X) = \begin{cases} \sum_i (x_i - \mu)^2 f_X(x_i), & \text{discrete} \\ \int_{-\infty}^{\infty} (x-\mu)^2 f_X(x)dx, & \textit{continuous} \end{cases}$$

$$\text{Standard deviation} \quad \sigma = \sqrt{\sigma^2}$$

20

Continuous Random Variables

- **Example (continue).** When X is normally distributed then its mean will be μ and its variance σ^2 (hence: standard deviation σ).
- Random variables with normally distribution are most important and widely used rv in applications
- The normal distribution is also called Gaussian distribution, named after Carl Friedrich Gauss.

21

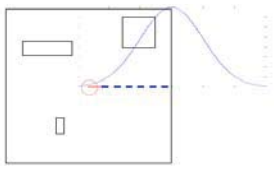
Robot Localization II

- Equipped with the knowledge of conditional probability, Bayes' Theorem, and normal distributions, we now return to the robot localization problem.
- Suppose that our robot has three range-finding sensors:
  - LEFT SENSOR points directly left
  - FORWARD SENSOR points directly ahead
  - RIGHT SENSOR points directly right
- Each sensor delivers a reading representing the distance to the next target (obstacle or edge of arena) in the sensor's direction.
- This reading is noisy: various readings are possible, clustered around the true value.

22

Robot Localization II

- It is often convenient (and reasonable) to assume that sensor readings will be normally distributed with mean equal to the true reading.
- **In our simulation**, we shall in fact assume that sensor readings will be approximately normally distributed with mean equal to the true reading and standard deviation equal to 1/3 of the true reading (never negative)



23

Robot Localization II

- Suppose now the robot polls one of its sensors, and obtains a reading. Let o be the event that that particular observation has been made, e.g.

  RIGHT SENSOR reports target at distance 13.
- According to Bayesian updating, the robot's new beliefs should be obtained by conditionaliziation on this event.
- That is, for all i, j, t in the range [0, 99], we want to set the new degree of belief in event $L_{i,j,t}$ to be

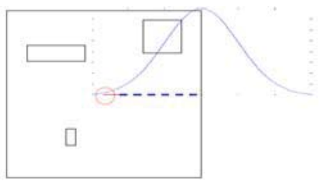
$$p_o(L_{i,j,t}) = p(L_{i,j,t} \mid o)$$

- How can we calculate this quantity?

Robot Localization II

- Since the robot knows what the environment is like, and since it can be equipped with (probabilistic) models of how its sensors work, it can compute the probability of any observed event o given a particular pose:

$$p(o \mid L_{i,j,t})$$

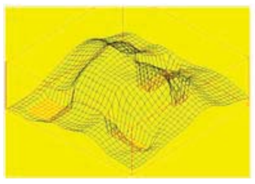


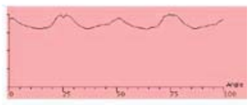
Robot Localization II

- Now use Bayes' theorem:

$$p(L_{i,j,t} \mid o) = \frac{p(o \mid L_{i,j,t})p(L_{i,j,t})}{\sum_{i',j',t'} p(o \mid L_{i',j',t'})p(L_{i',j',t'})}$$

- This will lead to revised degrees of belief about position and orientation, based on the revised probability distribution:





Robot Localization II

- The robot also needs to update its belief in response to its actions.
- The robot can perform 3 actions:
  - TURN LEFT (through 10/100 of a circle)
  - TURN RIGHT (through 10/100 of a circle)
  - GO FORWARD (by 10 units)
- We assume that turning actions are always faithfully executed.
- We assume that forward movements are also faithfully executed, except that the robot simply stops when it encounters an obstacle or the edge of the arena.
- In either case, we assume that the effects of actions are completely deterministic.

Robot Localization II

- Let a be the event that a certain action has been attempted (e.g. "Go forward 10 units").
- Let $L'_{i,j,t}$ be the event that the robot's new pose (subsequent to the action) is (i, j, t) .
- The robot can again use its knowledge of itself and its environment to obtain the conditional probabilities

$$p(L'_{i,j,t} \mid L_{i',j',t'} \wedge a)$$

for $i, j, t, i', j', t',$ in the range [0, 99].

Robot Localization II

- By the Extended Total Probability Formula:

$$p(L'_{i,j,t} \mid a) = \sum_{i',j',t'} p(L'_{i,j,t} \mid L_{i',j',t'} \wedge a)p(L_{i',j',t'} \mid a)$$

- Moreover, it is reasonable to assume:

$$p(L_{i',j',t'} \mid a) = p(L_{i',j',y'})$$

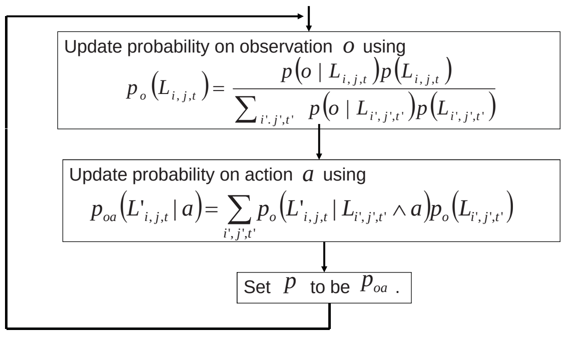
  After all: the probability that you are in a given location should not be affected by the action you subsequently perform*?
- Therefore:

$$p(L'_{i,j,t} \mid a) = \sum_{i',j',t'} p(L'_{i,j,t} \mid L_{i',j',t'} \wedge a)p(L_{i',j',t'})$$

- Thus, following an action a, the robot's probability of having pose (i, j, t) should be revised to $p(L'_{i,j,t} \mid a)$ as just calculated.
- To think about: Do you really believe the statement *?

Robot Localization II

- Thus, the robot's basic control loop is:



Update probability on observation o using

$$p_o(L_{i,j,t}) = \frac{p(o \mid L_{i,j,t})\, p(L_{i,j,t})}{\sum_{i',j',t'} p(o \mid L_{i',j',t'})\, p(L_{i',j',t'})}$$

Update probability on action a using

$$p_{oa}(L'_{i,j,t} \mid a) = \sum_{i',j',t'} p_o(L'_{i,j,t} \mid L_{i',j',t'} \wedge a)\, p_o(L_{i',j',t'})$$

Set p to be p_{oa} .

30

Robot Localization II

- This process works very well (for static environments)
- The problem of robot localization is, for such environments, effectively solved.

31

COMP14112: Artificial Intelligence Fundamentals

Lecture 3 - Foundations of Probabilistic Reasoning

Lecturer: Xiao-Jun Zeng

Email: x.zeng@manchester.ac.uk

Foundations of Probabilistic Reasoning

Outline

- Monty Hall Problem
- Dutch Book
- Reasoning of Dutch Book
- Reasoning of Monty Hall Problem

Last Lecture

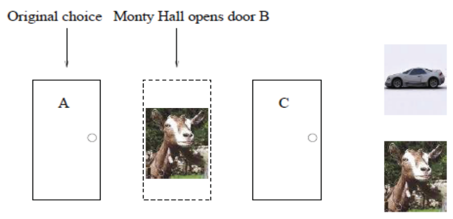
- In the last two lectures, we discuss the robot localization problem and show how to apply the probabilistic reasoning approach to determine the location of the robot.
- In this lecture, probabilistic reasoning approach is going to be further discussed by applying it to solve the Monty Hall paradox and avoid Dutch Book in betting.

Monty Hall Problem

- The Monty Hall problem is a probability puzzle based on the American television game show *Let's Make a Deal*, hosted by Monty Hall.
- In the game, you're given the choice of three doors:
  - Behind one door is a car;
  - behind the others, goats.
- You pick a door, say A, and the host, who knows what's behind the doors, opens another door, say B, which he knows has a goat. He then says to you, "Do you want to pick door C?"
- **Problem**. Is it to your advantage to switch your choice?

Monty Hall Problem

- The infamous Monty Hall paradox is as follows



- Now Monty Hall says "Do you want to pick door C?"
- **Question**: do you switch or stick?

Dutch Book

- Uncertainty can be understood in terms of betting.
- Let E be any event, e.g.

  Red Rum will win the Grand National tomorrow
- Consider the price (in £) you would consider fair for the following ticket:

£1	***if E***
£0	***if otherwise***

- This number is said to be your degree of belief in E.
- We shall assume that there always is a price you consider fair for such a ticket, between £0 and £1, and that this price is proportional to the stake.

Dutch Book

- What is the fair for the ticket:

$price = ?$

£1	***if E***
£0	***if otherwise***

$Fair\ \ price = £\, p(E)$

- Suppose that your degree of belief is p(E)=0.6=60% and you bet 10 times, then
  - Case 1. If the price is £0.5, you pay £0.5x10 times=£5 and you gain 60%x10 times x £1=£6. This is a too low price because you gain £1 and the bookmaker loses £1.
  - Case 1. If the price is £0.7, you pay £0.7x10 times=£7 and you gain 60%x10 times x £1=£6. This is a too high price because you lose £1 and the bookmaker gains £1.
  - CASE 3. If the price is £0.6, you pay £0.6x10 times=£6 and you gain 60%x10 times x £1=£6. This is a fair price as either you or the bookmaker does not gain or lose. This is why the fair price = £p(E).

6

Dutch Book

- Now what is the fair for the following ticket:

$price = ?$

£x	***if E***
£0	***if otherwise***

$Fair\ \ price = £x \times p(E)$

- Suppose that your degree of belief is p(E) and you bet n times, then
  - You pay is $price\ \ \times n$
  - you gain is $p(E) \times n \times £\, x$
  - To make the price be fair, you gain should be the same as you pay, that is,

$$\begin{aligned} price\ \ \times n\ &=\ p(E) \times n \times £\, x \\ \Rightarrow\ price\ \ &=\ £\, x \times\ p(E) \end{aligned}$$

7

Dutch Book

- A Dutch Book is
  - a sequence of bets, each of which the agent is disposed – given his degrees of belief – to accept, yet which taken together will cause the agent to lose money no matter what happens.
  - in other words, a Dutch Book is a set of bets, each of which you consider fair, that collectively guarantee your loss.

8

Dutch Book

- Let a and b be events, and suppose an agent adopts the following probabilities:

$$\begin{aligned} p(a) &= 0.6 \\ p(b) &= 0.5 \\ p\big(not\ (a \vee b)\big) &= 0.25 \\ p\big(not\ (a \wedge b)\big) &= 0.7 \end{aligned}$$

- the agent will then accept the following bets

bet 1:	£1 if a	60p
bet 2:	£1 if b	50p
bet 3:	£1 if $not\ (a \vee b)$	25p
Bet 4:	£1 if $not\ (a \wedge b)$	70p

9

Dutch Book

- The payoffs are:

a	b	bet 1 a	bet 2 b	bet 3 $not\ (a \vee b)$	bet 4 $not\ (a \wedge b)$	outcome
T	T	+40p	+50p	-25p	-70p	-5p
T	F	+40p	-50p	-25p	+30p	-5p
F	T	-60p	+50p	-25p	+30p	-5p
F	F	-60p	-50p	+75p	+30p	-5p

10

Reasoning of Dutch Book

- The above shows, the agent decides the prices based on his degree of belief. Unfortunately these prices are wrong as Dutch Book occurs and the agent will lose money no matter what happens.
- **Question**. What conditions need to be satisfied in order to avoid the Dutch Book?

11

Reasoning of Dutch Book

- **Question**: What conditions need to be satisfied in order to avoid the Dutch Book?
- **Answer** (Ramsey – de Finetti Theorem):
  - If the agent's degree of belief is a probability distribution function, i.e. it assigns a real number in [0,1] for each event and satisfies

    (i) $P(\Omega) = 1$ (K1)

    (ii) If $E1 \wedge E2 = \phi$ then

    $$p(E1 \vee E2) = p(E1) + p(E2) \quad \text{(K2)}$$
  - Otherwise, a Dutch Book can be constructed.

12

Reasoning of Dutch Book

- Now we analyse why the previous example leads to the Dutch book.
- Assume that the agent's degree of belief is a probability distribution

$$\begin{aligned} p(a) &= 0.6 \\ p(b) &= 0.5 \\ p(a \vee b) &= 1 - (p(not\,(a \vee b)) = 0.75 \\ p(a \wedge b) &= 1 - p(not\,(a \wedge b)) = 0.30 \\ \Rightarrow p(a \vee b) &= 0.75 \neq 0.8 = p(a) + p(b) - p(a \wedge b) \end{aligned}$$

- This is contradict to the property of a probability distribution:

$$p(a \vee b) = p(a) + p(b) - p(a \wedge b)$$

- In other words, the agent's degree of belief is not a probability distribution and this is why the Dutch Book occurs.

13

Reasoning of Dutch Book

- The Dutch book result suggests that an agent's degrees of belief should be represented as a probability distribution.
- There are many other ways of representing uncertainty in AI, but none has the mathematical and philosophical grounding enjoyed by probability theory.

14

Reasoning of Dutch Book

- Let E , F be any events.
- Consider the price (in £) you would consider fair for the following ticket:

£1	***if***	$E \wedge F$
£0	***if***	$not\ E \wedge F$
Money back	***if***	$not\ F$

- This number is said to be your **(conditional) degree of belief** in E given F , denoted $p_F(E)$.
- We shall assume that there always is a price you consider fair for such a ticket, between £0 and £1.

15

Reasoning of Dutch Book

- **Definition**: Given a probability distribution p and two events E and F with $p(F) > 0$, the conditional probability of E given F is defined by

$$p(E \mid F) = \frac{p(E \wedge F)}{p(F)}$$

- **Proposal**: If an agent's current degrees of belief match the probability distribution p , then his conditional degrees of belief should match his conditional probabilities:

$$p_F(E) = p(E \mid F)$$

whenever $p(F) > 0$.

16

Reasoning of Dutch Book

- The following (synchronic) Dutch book justifies this proposal.
- Suppose an agent takes the conditional bet

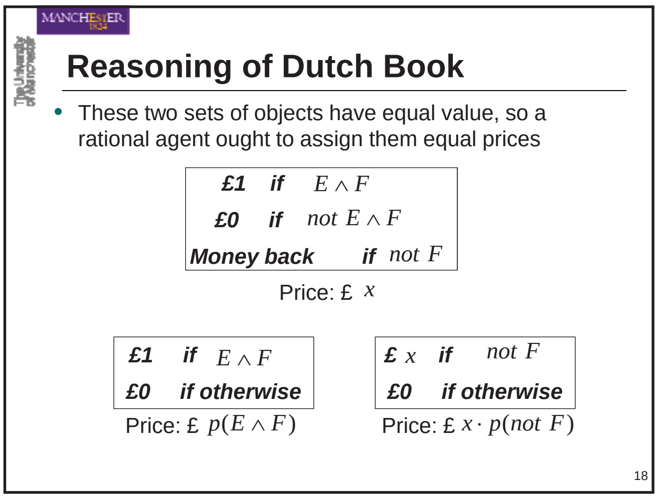
£1	***if***	$E \wedge F$
£0	***if***	$not\ E \wedge F$
Money back	***if***	$not\ F$

- to be worth £x. Now compare this ticket with the following pair of tickets:.

£1	***if*** $E \wedge F$
£0	***if otherwise***

£x	***if*** $not\ F$
£0	***if otherwise***

17



Reasoning of Dutch Book

- These two sets of objects have equal value, so a rational agent ought to assign them equal prices

£1	***if***	$E \wedge F$
£0	***if***	$not\ E \wedge F$
Money back	***if***	$not\ F$

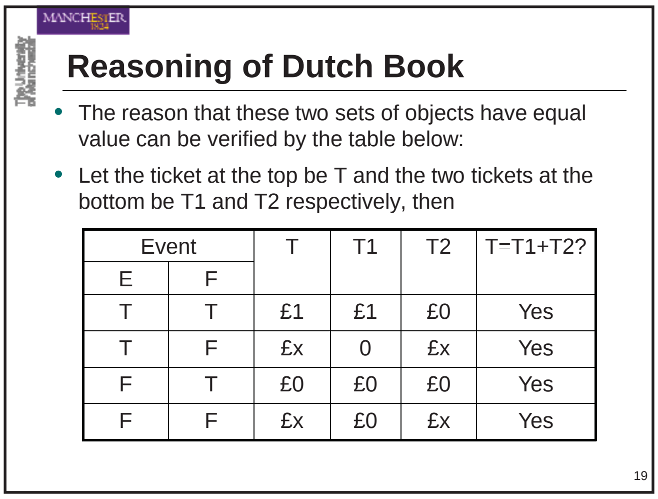
Price: £ x

£1	***if***	$E \wedge F$
£0	***if otherwise***	

Price: £ $p(E \wedge F)$

£ x	***if***	$not\ F$
£0	***if otherwise***	

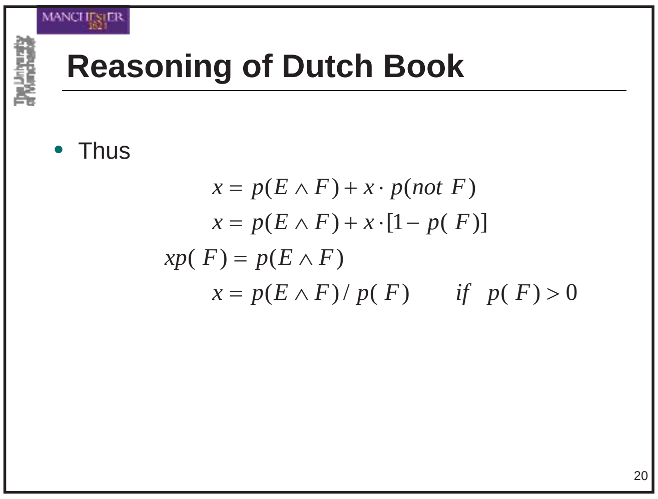
Price: £ $x \cdot p(not\ F)$



Reasoning of Dutch Book

- The reason that these two sets of objects have equal value can be verified by the table below:
- Let the ticket at the top be T and the two tickets at the bottom be T1 and T2 respectively, then

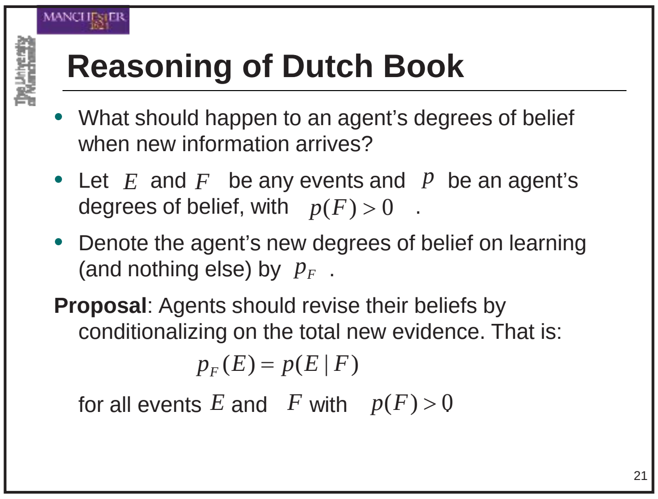
Event		T	T1	T2	T=T1+T2?
E	F				
T	T	£1	£1	£0	Yes
T	F	£x	0	£x	Yes
F	T	£0	£0	£0	Yes
F	F	£x	£0	£x	Yes



Reasoning of Dutch Book

- Thus

$$
\begin{aligned}
x &= p(E \wedge F) + x \cdot p(not\ F) \\
x &= p(E \wedge F) + x \cdot [1 - p(F)] \\
xp(F) &= p(E \wedge F) \\
x &= p(E \wedge F) / p(F) \qquad if\ \ p(F) > 0
\end{aligned}
$$



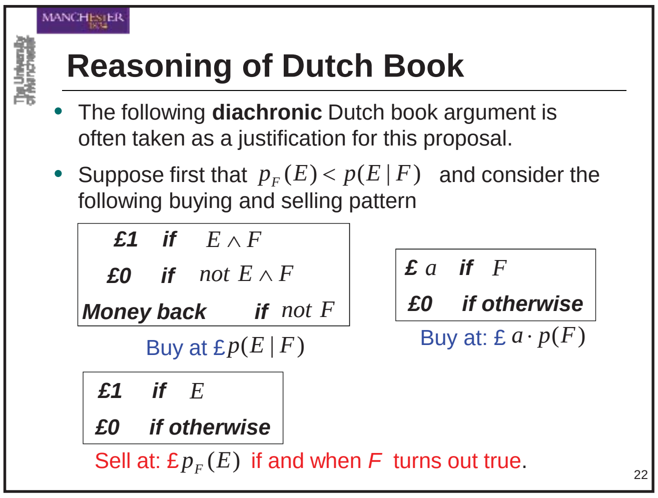
Reasoning of Dutch Book

- What should happen to an agent's degrees of belief when new information arrives?
- Let E and F be any events and p be an agent's degrees of belief, with $p(F) > 0$.
- Denote the agent's new degrees of belief on learning (and nothing else) by p_F .

Proposal: Agents should revise their beliefs by conditionalizing on the total new evidence. That is:

$$p_F(E) = p(E \mid F)$$

for all events E and F with $p(F) > 0$



Reasoning of Dutch Book

- The following **diachronic** Dutch book argument is often taken as a justification for this proposal.
- Suppose first that $p_F(E) < p(E \mid F)$ and consider the following buying and selling pattern

£1	***if***	$E \wedge F$
£0	***if***	$not\ E \wedge F$
Money back	***if***	$not\ F$

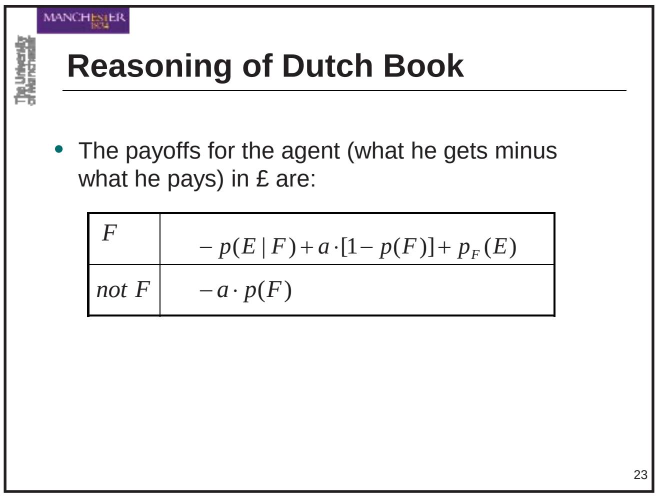
Buy at £$p(E \mid F)$

£ a	***if***	F
£0	***if otherwise***	

Buy at: £ $a \cdot p(F)$

£1	***if***	E
£0	***if otherwise***	

Sell at: £$p_F(E)$ if and when F turns out true.



Reasoning of Dutch Book

- The payoffs for the agent (what he gets minus what he pays) in £ are:

F	$-p(E \mid F) + a \cdot [1 - p(F)] + p_F(E)$
$not\ F$	$-a \cdot p(F)$

Reasoning of Dutch Book

- Let the two tickets at the top be T1 and T2 and the ticket at the bottom as T3, then the payoffs for the agent can be verified as below:

Event		Ticket 1		Ticket 2		Ticket 3		Total
E	F	Gain	Pay	Gain	Pay	Gain	Pay	
T	T	+1	- p(E|F)	+a	-ap(F)	-1	+P_F(E)	- p(E|F)+a-ap(F)+P_F(E)
F	T	0	- p(E|F)	+a	-ap(F)	0	+P_F(E)	- p(E|F)+a-ap(F)+P_F(E)
T	F	+p(E|F)	- p(E|F)	+0	-ap(F)	N/A	N/A	-ap(F)
F	F	+p(E|F)	- p(E|F)	+0	-ap(F)	N/A	N/A	-ap(F)

Reasoning of Dutch Book

- The latter payoff is negative provided $a > 0$; the former is also negative if

$$-p(E\,|\,F) + a\cdot[1 - p(F)] + p_F(E) < 0$$

i.e.,

$$a\cdot[1 - p(F)] < p(E\,|\,F) - p_F(E)$$

i.e., (assuming $p(F) \neq 1$) if

$$a < [p(E\,|\,F) - p_F(E)]/[1 - p(F)]$$

- Thus, if the agent has $p(E\,|\,F) > p_F(E)$, we choose any a such that $0 < a < [p(E\,|\,F) - p_F(E)]/[1 - p(F)]$, and the agent loses in every case.
- If the agent has $p(E\,|\,F) < p_F(E)$, reverse buying and selling instructions.

Reasoning of Dutch Book

- Hence, if $p(E\,|\,F) \neq p_F(E)$, we can make money out of the agent using a **diachronic Dutch book**
- Hence (so the argument goes), rational agents satisfy

$$p(E\,|\,F) = p_F(E)$$

- That is: they respond to new evidence by performing **Bayesian updating**.

Reasoning of Monty Hall Problem

- You have to be careful with Bayesian updating.
- Recall the Monty Hall puzzle, and let A stand for the proposition "The car is behind door A" (similarly, with B and C).
- Note that

$$p[A \wedge (not\ B)] = p(A) = 1/3$$
$$p(not\ B) = 1 - p(B) = 2/3$$
$$p(A\,|\,not\ B) = p[A \wedge (not\ B)]/p(not\ B) = 1/2$$

- Gulp! This suggests that it does not matter whether we switch or not!
- Yet we know that switchers have 2/3 chance to win the car than non-switchers.

Reasoning of Monty Hall Problem

- The answer to the puzzle involves getting the 'right' representation.
- Here a solution is going to give to show how to get the right representation

Reasoning of Monty Hall Problem

Solution: Let M-B stand for "Monty Hall opens door B", then

- Under event A, $p(M_B\,|\,A) = 1/2$ as Monty Hall can open B or C;
- Under event B, $p(M_B\,|\,B) = 0$ as Monty Hall only opens the no-car door;
- Under event C, $p(M_B\,|\,C) = 1$ as Monty Hall can not opens door C and has to open door B;
- Now apply Bayes' theorem (alternative form):

$$P(A|M_B) = \frac{P(M_B\,|\,A)p(A)}{P(M_B\,|\,A)P(A) + P(M_B\,|\,B)P(B) + P(M_B\,|\,C)P(C)} = \frac{(1/2)\times(1/3)}{(1/2)\times(1/3) + 0\times(1/3) + 1\times(1/3)} = \frac{1}{3}$$

$$P(B|M_B) = \frac{P(M_B\,|\,B)p(B)}{P(M_B\,|\,A)P(A) + P(M_B\,|\,B)P(B) + P(M_B\,|\,C)P(C)} = \frac{0\times(1/3)}{(1/2)\times(1/3) + 0\times(1/3) + 1\times(1/3)} = 0$$

$$P(C|M_B) = \frac{P(M_B\,|\,C)p(C)}{P(M_B\,|\,A)P(A) + P(M_B\,|\,B)P(B) + P(M_B\,|\,C)P(C)} = \frac{1\times(1/3)}{(1/2)\times(1/3) + 0\times(1/3) + 1\times(1/3)} = \frac{2}{3}$$

Reasoning of Monty Hall Problem

- This reason to get the conclusion that the stick and the switch has the same probability to win is due to believing that the event not B (i.e., the car is not behind B) is the same event as M-B (i.e., “Monty Hall opens door B”).
- But they are different. The former is a pure random event but the latter is not as Monty Hall only opens the “no-car” door.
- This can be seen as the probability for event “Not B” is 2/3 whereas the probability for event “M-B” is 1/2.
- Therefore You have to be careful with Bayesian updating.

30

Probabilistic Reasoning

You have been warned!

31

COMP14112: Artificial Intelligence Fundamentals

Lecture 4 – Overview and Brief History of AI

Lecturer: Xiao-Jun Zeng

Email: x.zeng@manchester.ac.uk

Lecture 4- Introduction to AI

Outline

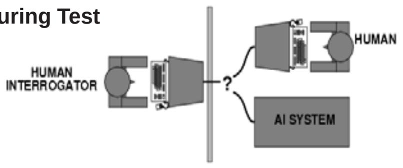
- **What is AI**
- **Brief history of AI**
- **AI Problems and Applications**

What is AI

It's a lot of different things to a lot of different people:

- **Computational models of human behaviour**
  - Programs that behave (externally) like humans.
  - This is the original idea from Turing and the well known Turing Test is to use to verify this

Turing Test



What is AI

It's a lot of different things to a lot of different people:

- **Computational models of human "thought"**
  - Programs that operate (internally) the way humans do
- **Computational systems that behave intelligently?**
  - But what does it mean to behave intelligently?
- **Computational systems that behave rationally**
  - More widely accepted view

What is AI

- **What means "behave rationally" for a person/system:**
  - Take the right/ best action to achieve the goals, based on his/its knowledge and belief
- **Example**. Assume I don't like to get wet (my goal), so I bring an umbrella (my action). Do I behave rationally?
  - The answer is dependent on my knowledge and belief
  - If I've heard the forecast for rain and I believe it, then bringing the umbrella is rational.
  - If I've not heard the forecast for rain and I do not believe that it is going to rain, then bringing the umbrella is not rational.

What is AI

<u>Note on behave rationally or rationality</u>

- **"Behave rationally" does not always achieve the goals successfully**
  - Example.
    - My goals – (1) do not get wet if rain; (2) do not be looked stupid (such as bring an umbrella when no raining)
    - My knowledge/belief – weather forecast for rain and I believe it
    - My rational behaviour – bring an umbrella
    - The outcome of my behaviour: If rain, then my rational behaviour achieves both goals; If not rain, then my rational behaviour fails to achieve the 2nd goal
- **The successfulness of "behave rationally" is limited by my knowledge and belief**

What is AI

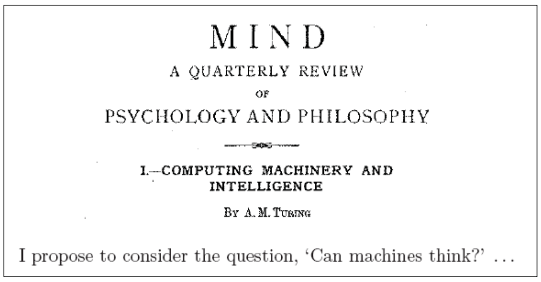
Note on behave rationally or rationality

- **Another limitation of “behave rationally” is the ability to compute/ find the best action**
  - **In chess-playing, it is sometimes impossible to find the best action among all possible actions**
- **So, what we can really achieve in AI is the limited rationality**
  - Acting based to your best knowledge/belief (best guess sometimes)
  - Acting in the best way you can subject to the computational constraints that you have

6

Brief history of AI

- The history of AI begins with the following articles:
  - Turing, A.M. (1950), Computing machinery and intelligence, Mind, Vol. 59, pp. 433-460.


MIND
A QUARTERLY REVIEW
OF
PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE
By A. M. Turing

I propose to consider the question, ‘Can machines think?’ . . .

7

Alan Turing - Father of AI

Alan Turing (OBE, FRS)

- Born 23 June 1912, Maida Vale, London, England
- Died 7 June 1954 (aged 41), Wilmslow, Cheshire, England
- Fields: Mathematician, logician, cryptanalyst, computer scientist
- Institutions:
  - University of Manchester
  - National Physical Laboratory
  - Government Code and Cypher School (Britain's codebreaking centre)
  - University of Cambridge

Alan Turing memorial statue in Sackville Park, Manchester

8

Turing’s paper on AI

- You can get this article for yourself: go to http://www.library.manchester.ac.uk/eresources/

  select ‘Electronic Journals’ and find the journal Mind. The reference is:
  - A. M. Turing, “Computing Machinery and Intelligence”, Mind, (New Series), Vol. 59, No. 236, 1950, pp. 433-460.
- You should read (and make notes on) this article in advance of your next Examples class!

9

Brief history of AI - The Birth of AI

- **The birth of artificial intelligence**
  - 1950: Turing’s landmark paper “Computing machinery and intelligence” and Turing Test
  - 1951: AI programs were developed at Manchester:
    - A draughts-playing program by Christopher Strachey
    - A chess-playing program by Dietrich Prinz
    - These ran on the Ferranti Mark I in 1951.
  - 1955: Symbolic reasoning and the Logic Theorist
    - Allen Newell and (future Nobel Laureate) Herbert Simon created the "Logic Theorist". The program would eventually prove 38 of the first 52 theorems in Russell and Whitehead's Principia Mathematica
  - 1956: Dartmouth Conference - "Artificial Intelligence" adopted

10

Brief history of AI - The Birth of AI

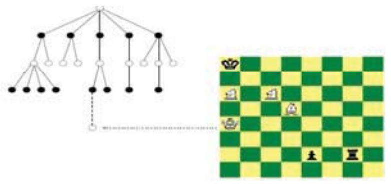
- **The birth of artificial intelligence**
  - 1956: Dartmouth Conference - "Artificial Intelligence" adopted
  - The term ‘Artificial Intelligence’ was coined in a proposal for the conference at Dartmouth College in 1956

  - The term stuck, though it is perhaps a little unfortunate . . .

11

Brief history of AI – The Birth of AI

- One of the early research in AI is search problem such as for game-playing. Game-playing can be usefully viewed as a search problem in a space defined by a fixed set of rules



- Nodes are either white or black corresponding to reflect the adversaries' turns.
- The tree of possible moves can be searched for favourable positions.

Brief history of AI – The Birth of AI

- The real success of AI in game-playing was achieved much later after many years' effort.
- It has been shown that this search based approach works extremely well.
- In 1996 IBM Deep Blue beat Gary Kasparov for the first time. and in 1997 an upgraded version won an entire match against the same opponent.

Brief history of AI – The Birth of AI

- Another of the early research in AI was applied the similar idea to **deductive logic**:

All men are mortal	$\forall$ x (man(x) -> mortal(x))
Socrates is a man	man(Socrates)
Socrates is mortal	mortal(Socrates)

- The discipline of developing programs to perform such logical inferences is known as (automated) **theorem-proving**
- Today, theorem-provers are highly-developed . . .

Brief history of AI – The Birth of AI

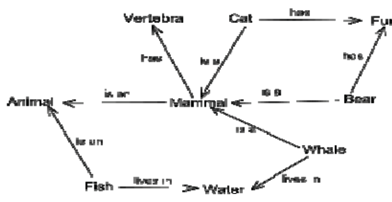
- In the early days of AI, it was conjectured that theorem-proving could be used for commonsense reasoning
- The idea was to code common sense knowledge as logical axioms, and employ a theorem-prover.
- Early proponents included John McCarthy and Patrick Hayes.
- The idea is now out of fashion: logic seems to rigid a formalism to accommodate many aspects of commonsense reasoning.
- Basic problem: such systems do not allow for the phenomenon of uncertainty.

Brief history of AI - Golden years 1956-74

- **Research**:
  - **Reasoning as search:** Newell and Simon developed a program called the "General Problem Solver".
  - **Natural language Processing**: Ross Quillian proposed the semantic networks and Margaret Masterman & colleagues at Cambridge design semantic networks for machine translation
  - **Lisp**: John McCarthy (MIT) invented the Lisp language.
- **Funding for AI research**:
  - Significant funding from both USA and UK governments
- **The optimism**:
  - 1965, Simon: "machines will be capable, within twenty years, of doing any work a man can do
  - 1970, Minsky: "In from three to eight years we will have a machine with the general intelligence of an average human being."

Brief history of AI - The golden years

- Semantic Networks
  - A semantic net is a network which represents semantic relations among concepts. It is often used as a form of knowledge representation.
  - Nodes : used to represent objects and descriptions.
  - Links : relate objects and descriptors and represent relationships.



Brief history of AI - The golden years

- Lisp
  - Lisp (or LISP) is a family of computer programming languages with a long history and a distinctive, fully parenthesized syntax.
  - Originally specified in 1958, Lisp is the second-oldest high-level programming language in widespread use today; only Fortran is older.
  - LISP is characterized by the following ideas:
    - computing with symbolic expressions rather than numbers
    - representation of symbolic expressions and other information by list structure in the memory of a computer
    - representation of information in external media mostly by multi-level lists and sometimes by S-expressions
  - An example: lisp S-expression:

```
(+ 1 2 (IF (> TIME 10) 3 4))
```

18

Brief history of AI - The first AI winter

- **The first AI winter 1974−1980:**
  - **Problems**
    - **Limited computer power**: There was not enough memory or processing speed to accomplish anything truly useful
    - **Intractability and the combinatorial explosion**. In 1972 Richard Karp showed there are many problems that can probably only be solved in exponential time (in the size of the inputs).
    - **Commonsense knowledge and reasoning**. Many important applications like vision or natural language require simply enormous amounts of information about the world and handling uncertainty.
  - **Critiques from across campus**
    - Several philosophers had strong objections to the claims being made by AI researchers and the promised results failed to materialize
  - **The end of funding**
    - The agencies which funded AI research became frustrated with the lack of progress and eventually cut off most funding for AI research.

19

Brief history of AI - Boom 1980–1987

- **Boom 1980–1987:**
  - In the 1980s a form of AI program called "expert systems" was adopted by corporations around the world and knowledge representation became the focus of mainstream AI research
    - The power of expert systems came from the expert knowledge using **rules** that are derived from the domain experts
    - In 1980, an expert system called XCON was completed for the Digital Equipment Corporation. It was an enormous success: it was saving the company 40 million dollars annually by 1986
    - By 1985 the market for AI had reached over a billion dollars
  - The money returns: the fifth generation project
    - Japan aggressively funded AI within its fifth generation computer project (but based on another AI programming language - Prolog created by Colmerauer in 1972)
    - This inspired the U.S and UK governments to restore funding for AI research

20

Brief history of AI - Boom 1980–1987

- The expert systems are based a more flexibly interpreted version of the 'rule-based' approach for knowledge representation to replace the logic representation and reasoning

  If <conditions> then <action>

- Collections of (possibly competing) rules of this type are sometimes known as production-systems
  - This architecture was even taken seriously as a model of Human cognition
  - Two of its main champions in this regard were Allen Newell and Herbert Simon.

21

Brief history of AI - Boom 1980–1987

- One of the major drawbacks of rule-based systems is that they typically lack a clear semantics

  If C then X

  If D then Y

  . . .

  Okay, so now what?

- It is fair to say that this problem was never satisfactorily resolved.
- Basic problem: such systems fail to embody any **coherent underlying theory** of uncertain reasoning, and they were difficult to update and could not learn.

22

Brief history of AI - the second AI winter

- **the second AI winter 1987−1993**
  - In 1987, the Lisp Machine market was collapsed, as desktop computers from Apple and IBM had been steadily gaining speed and power and in 1987 they became more powerful than the more expensive Lisp machines made by Symbolics and others
  - Eventually the earliest successful expert systems, such as XCON, proved too expensive to maintain, due to difficult to update and unable to learn.
  - In the late 80s and early 90s, funding for AI has been deeply cut due to the limitations of the expert systems and the expectations for Japan's Fifth Generation Project not being met
  - **Nouvelle AI:** But in the late 80s, a completely new approach to AI, based on robotics, has bee proposed by Brooks in his paper "Elephants Don't Play Chess", based on the belief that, to show real intelligence, a machine needs to have a body — it needs to perceive, move, survive and deal with the world.

23

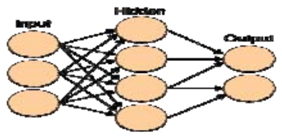
Brief history of AI - AI 1993−present

- AI achieved its greatest successes, albeit somewhat behind the scenes, due to:
  - the incredible power of computers today
  - a greater emphasis on solving specific subproblems
  - the creation of new ties between AI and other fields working on similar problems
  - a new commitment by researchers to solid mathematical methods and rigorous scientific standards, in particular, based probability and statistical theories
  - Significant progress has been achieved in neural networks, probabilistic methods for uncertain reasoning and statistical machine learning, machine perception (computer vision and Speech), optimisation and evolutionary computation, fuzzy systems, Intelligent agents.

24

Artificial Neural Networks (ANN) Approach

- Mathematical / computational model that tries to simulate the structure and/or functional aspects of biological neural networks



- Such networks can be used to learn complex functions from examples.

25

Probabilistic and Statistical Approach

- The rigorous application of probability theory and statistics in AI generally gained in popularity in the 1990s and are now the dominant paradigm in:
  - Machine learning
  - Pattern recognition and machine perception, e.g.,
    - Computer vision
    - Speech recognition
  - Robotics
  - Natural language processing

26

AI Problems and Applications today

- Deduction, reasoning, problem solving such as
  - Theorem-provers, solve puzzles, play board games
- Knowledge representation such as
  - Expert systems
- Automated planning and scheduling
- Machine Learning and Perception such as
  - detecting credit card fraud, stock market analysis, classifying DNA sequences, speech and handwriting recognition, object and facial recognition in computer vision

27

AI Problems and Applications today

- Natural language processing such as
  - Natural Language Understanding
  - Speech Understanding
  - Language Generation
  - Machine Translation
  - Information retrieval and text mining
- Motion and manipulation such as
  - Robotics to handle such tasks as object manipulation and navigation, with sub-problems of localization (knowing where you are), mapping (learning what is around you) and motion planning (figuring out how to get there)
- Social and business intelligence such as
  - Social and customer behaviour modelling

28

What Next

- This is the end of Part 1 of Artificial Intelligence Fundamentals, which includes
  - Robot localization
  - Overview and brief history of AI
  - Foundations of probability for AI
- What next:
  - You listen to Dr. Tim Morris telling you how to use what you have learned about probability theory to do automated speech recognition
- Finally
  - There will be a revision lecture of Part 1 in Week 10
  - And Thank you!

29

COMP14112

Artificial Intelligence Fundamentals

Comp14112: Lab 1
Robot Localization

Second Semester
2013-2014

School of Computer Science
University of Manchester

Comp14112: Lab 1
Robot Localization

Xiao-Jun Zeng

Academic session: 2013 - 2014

1 Introduction

This lab involves writing the critical pieces of code for the probabilistic robot localization algorithm demonstrated in the lectures. The code can be found in the directory

`/opt/info/courses/COMP14112/labs/lab1/robot`.

or by going to the course webpage

`http://www.cs.manchester.ac.uk/ugt/2011/COMP14112/`

and following the links.

You should work in your `$HOME/COMP14112/ex1` directory and make a copy of the entire robot directory there. Ensure that it has the name *robot*, and adjust your `CLASSPATH` variable if necessary so that it includes the `$HOME/COMP14112/ex1` directory. (That way, the Java compiler will be able to see the files.) These files feature some methods which have been replaced with dummy versions that stop the simulator working properly. Your job is to rewrite these methods so that they contain correctly working code.

2 Getting started

Change directory to `$HOME/COMP14112/ex1` and verify that you can compile and run the code:

```
> javac robot/RunRobot.java
> java robot.RunRobot
```

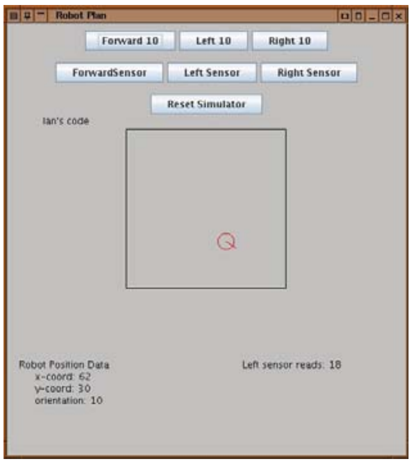
You should see everything working just as in the lectures. The only snag is that the graphs representing the robot's degrees of belief have gone flat! This is because `RobotBeliefState.java` contains only dummy versions of the critical methods

```
initializeMatrices()
updateProbabilityOnObservation(Observation)
updateProbabilityOnAction(Action),
```

responsible for determining the robot's degrees of belief. (The dummy versions just assign silly values, which make no sense.)

To help you see what the program is actually supposed to do, however, we have provided the `.class` file for the class `SolutionRobotBeliefState`, which is a (final) subclass of `RobotBeliefState`. It has correctly functioning versions of the above three methods which override the dummy versions in `RobotBeliefState`. (It also prints a disclaimer in the main window saying that this is not your code!)

To run the correctly functioning version, change the code snippet



```
Ian's code

Robot Position Data
    x-coord: 62
    y-coord: 30
    orientation: 10
                                    Left sensor reads: 18
```

Figure 1: The robot in its arena, with pose (i, j, t)

```
    // Make robot's mind using the map
    // RobotBeliefState r= new SolutionRobotBeliefState(m);
    RobotBeliefState r= new RobotBeliefState(m);
```

in `RunRobot.java` so that it reads

```
    // Make robot's mind using the map
    RobotBeliefState r= new SolutionRobotBeliefState(m);
    // RobotBeliefState r= new RobotBeliefState(m);,
```

recompile, and run. You should see the probability graphs functioning as normal.

Of course, the point of the exercise is that you write your own fully functioning versions of the above methods in the file `RobotBeliefState.java`. (Thus, your final program should not use `SolutionRobotBeliefState` at all.)

3 Guide to the code

The robot inhabits a square arena cluttered with a number of rectangular obstacles. At any time, it occupies a position (i, j), and has an orientation t, where i, j and t are numbers in the range 0 to `RunRobot.SIZE`-1. (The constant `RunRobot.SIZE` is currently set to 100.) The position (i, j) is understood in the usual (Cartesian) way, and the orientation t is interpreted as the forward-facing direction of the robot in units of 2π/`RunRobot.SIZE`, measured *clockwise* in radians from the x-axis (Fig. 1). The triple (i, j, t) of position and orientation together is known as the robot's *pose*. When you run the program `RunRobot`, you will see a window entitled `Robot Plan` displaying the robot's pose both graphically and textually.

Let us consider how actions are dealt with in our simulator. In the real world, the effects of an action depend on factors about which there is some uncertainty: how run-down the batteries are, how much the wheels slip on the floor, and so on. That is to say: actions are *noisy*. The simulator has a facility to model this noise—albeit in a simplified way—which is turned on (or off) by depressing the button `Toggle Prob. Actions`. (A text string in the lower left-hand corner of the `Robot Plan` window indicates whether this feature is turned on or off.) The default is for this facility to be turned *off*, so that actions are assumed to be *deterministic*: given the robot's current pose, there is no uncertainty about the effect of any given action.

Let us consider the deterministic case first. The robot is assumed, for the purposes of this simulation, to be a point-object, and to turn on its own axis. The robot has three actions which it can perform: move forward (by 10 units); turn right (by 10 units) and turn left (by 10 units). We refer to the number 10 as the *parameter* of the action. Clicking one of the buttons `Left 10`, `Forward 10` or `Right 10` causes the action in question to be performed. Because we are assuming actions to be deterministic, these actions are always assumed to be performed with complete accuracy. In particular, since the robot is modelled as a point-object, turning actions are always executed normally: pressing `Left 10` always decrements the robot's orientation by exactly 10 units, and similarly for `Right 10`. With forward moves, however, there is a complication. On occasions, a forward move of 10 units may be impossible, because it would result in a collision. In that case, the robot merely goes as far forward as it can. Thus, if the distance from the robot to the next obstacle (or perimeter of the arena) is d units in the direction it is facing, pressing `Left 10` always results in a forward move of either d units or 10 units, whichever is the smaller. Warning: in the graphical display, the robot is depicted as a red circle of non-zero size: the periphery of this circle may overlap obstacles or the edge of the arena; however, its centre should never do so.

When the probabilistic actions feature is turned on, the simulator needs to model the uncertain effects of actions. The strategy adopted here is to subject the *parameter* of any action to a certain amount of random variation. That is, instead of simply taking the parameter to be 10, we take it to be sampled from a random distribution with *mean* 10 and standard deviation equal to 3.33. (This has the effect that almost all of the probability mass lies between 0 and 20.) For example, suppose that the `Left 10` button is pressed. The simulator generates a random number from the distribution just described—say 13—and then executes a left-hand turn of 13 units. Likewise, suppose that the `Forward 10` button is pressed. The simulator again generates a random number from the distribution just described—say 9—and then tries to execute a forward move of 9 units. I say *tries to*, because the robot may bump into an obstacle after, say, 6 units of motion. In that case, the robot simply goes as far as it can and stops, exactly as in the deterministic case.

Next we consider how sensors are modelled. The robot has three sensors: one pointing left (at $\pi/2$ to the robot's orientation), another pointing forward, and a third pointing right. The sensors return a value representing the distance measured in the direction the sensor is pointing to the nearest target (either the edge of the arena or an obstacle). The measurement is *noisy*. Clicking one of the buttons `Left sensor`, `Forward sensor` or `Right sensor` causes the sensor in question to be polled. The window `Robot Plan` displays the values actually returned, so that you can see the information that the robot gets.

The overall structure of the code is as follows. Everything is contained in a package called `robot`. The main class, `RunRobot`, creates an instance `s` of the class `Scenario` to represent the actual (current) state of the world, and an instance `r` of the class `RobotBeliefState` to represent the robot's degrees of belief about the world. Look at the code in `RunRobot.java`, and make sure that you can find the relevant lines of code. You should keep these two aspects of the simulator separate in your mind: on the one hand, the scenario `s` stores the actual state of the world, about which the simulator—but not the robot—has perfect information; on the other, the robot's belief-state `r` stores the robot's beliefs about the world, which arise from the robot's limited observations using noisy sensors.

The class `Scenario` has fields `map` and `robotPose`, which represent the static environment and the robot's current pose, respectively. The field `map` is of class `WorldMap` and the field `robotPose` is of class `Pose`. Look at the code in `WorldMap.java` and `Pose.java`, as well as the Javadoc files, and make sure you understand the basic structure. Notice in particular that `WorldMap` and `Scenario` have a number of methods to return useful information about the robot in its environment. For example, the `WorldMap` method

```
public double getMinDistanceToTarget(int x, int y, int t)
```

returns the distance to the nearest target (either the border of the arena or an obstacle) in the forward-facing direction of the robot, assuming that it has pose (x, y, t). At start-up, the `main` method in `RunRobot` creates `s.map` and populates it with obstacles. It also initializes `s.pose` to (50,50,0), representing the mid-point of the arena, facing directly right.

The class `RobotBeliefState` has a member variable `beliefMatrix`, of type `double[][][]`. When `r` is created, `beliefMatrix` is set to a 3-dimensional cubic matrix of size `RunRobot.SIZE`. The cell `beliefMatrix[`i`][`j`][`t`]` represents the robot's degree of belief that its current pose is (i, j, t).

Clicking on an action button in the window `Robot Plan` affects the scenario `s` on the one hand, and the robot's belief-state `r` on the other. The effect on `s` is to execute the method `s.executeAction(`a`)`, where a is an object of class `Action` representing the action in question. The class `Action` has two fields: `type`, which encodes the type of action performed (`TURN_LEFT`, `GO_FORWARD` or `TURN_RIGHT`), and `parameter`, which gives the size of the movement in question. The job of `s.executeAction(`a`)` is simply to update the robot's pose in response to the action performed. The effect on `r` of clicking on an action button is to cause the method `r.updateProbabilityOnAction(Action `a`)` to be executed, which revises the robot's degrees of belief given that the action a has been performed. The details of this method are discussed in greater detail below.

Clicking on a sensor button in the window `Robot Plan` causes the method `s.pollSensor(`d`)` to be executed, where d is an object of the enumeration class `SensorType`, representing the sensor polled. This method returns an object of class `Observation`, representing the observation made. The class `Observation` has two fields: `sensor`, which identifies the sensor involved (`LEFT_SENSOR`, `FORWARD_SENSOR` or `RIGHT_SENSOR`), and `reading`, which gives the reading returned by that sensor (as a non-negative integer). Once `s.pollSensor(`d`)` has returned the observation—say, o—the method
`r.updateProbabilityOnObservation(Observation `o`)` is executed. This method revises the robot's degrees of belief in response to the information contained in o. The details of this method are discussed in greater detail below.

It is difficult to display a 3-dimensional array graphically. However, we can display the robot's degrees of belief about its position and orientation separately; and this is done in the windows `Robot Position Graph` and `Robot Orientation Graph`, respectively. If we write $l_{i,j,t}$ for the proposition "The robot has pose (i, j, t)", $l_{i,j}$ for the proposition "The robot has position (i, j)", and l_t for the proposition "The robot has orientation t", then the probabilities for the latter two can be derived from the those of the first as follows:

$$p(l_{i,j}) = \sum_t p(l_{i,j,t}) \tag{1}$$

$$p(l_t) = \sum_i \sum_j p(l_{i,j,t}), \tag{2}$$

where all summations run from 0 to `RunRobot.SIZE`-1.

4 The tasks

You have four tasks. The first two are relatively routine, the third is a little harder, and the fourth very tough. There is no shame at all in failing to do the last exercise: it is worth only 4 marks (20% of the marks), and I've put it there for students who really want a programming challenge.

1. (4 marks) Modify the code for the `RobotBeliefState` method

```
public void initializeMatrices(),
```

so that the member variable `beliefMatrix[][][]` is initialized to a flat probability distribution over those poses corresponding to *unoccupied squares*. (Poses corresponding to occupied squares must have probability zero). Note that you will have to work out *how many* poses correspond to unoccupied squares, so that you can 'normalize' the entries in `beliefMatrix[][][]` (i.e. make them all sum to 1).

Hint: the `WorldMap` method `isOccupied(int` i`, int` j`)` returns true if, in the world in question, the position (i, j) is occupied by an object; note that this method crashes if (i, j) are not integers between 0 and `RunRobot.SIZE`-1.

Warning: do not confuse *poses* with *positions*!

2. (6 marks) Modify the code for the `RobotBeliefState` method

   `public void updateProbabilityOnObservation(Observation` o`)`

   so that the array `beliefMatrix[][][]` is updated by conditioning on the observation o.

   Suppose that the object `r` of class `RobotBeliefState` represents the robot's state of mind. And let $l_{i,j,t}$ denote the proposition that the robot is in pose (i, j, t). Then the entry

   `r.beliefMatrix[`i`][`j`][`t`]`

   stores the probability $p(l_{i,j,t})$ of this proposition. We want to update each such entry so that it stores the conditional probability

$$p(l_{i,j,t}|o) = \frac{p(o|l_{i,j,t})p(l_{i,j,t})}{\sum_{i'}\sum_{j'}\sum_{t'} p(o|l_{i',j',t'})p(l_{i',j',t'})}, \tag{3}$$

   where the sum ranges over all (i', j', t') in the range 0 to `RunRobot.SIZE`-1. To compute the right-hand side of Equation (3), use the `WorldMap` method

   `getObservationProbability(Pose` p`, Observation` o`),`

   which returns the conditional probability $p(o|l_{i,j,t})$, provided that p is the pose (i, j, t).

   Hint: To call `getObservationProbability(Pose` p`, Observation` o`)`, you will need an object of class `Pose` to pass to it. If you make many such calls in a loop, don't create a new `Pose`-object every time you go through the loop!

   Hint: The `RobotBeliefState` member variable `workMatrix` is a temporary matrix, with the same dimensions as `beliefMatrix`. You may find it handy.

3. (6 marks) Modify the code for the `RobotBeliefState` method

   `public void updateProbabilityOnAction(Action` a`),`

   so that the array `beliefMatrix[][][]` is updated to reflect the fact that the robot has performed action a. Remember from the lectures how this is done. Writing $l_{i',j',t'}$ for "The robot's pose *before* the action is (i', j', t')", writing $l'_{i,j,t}$ for "The robot's pose *after* the action is (i, j, t)", and writing a for "The action that was performed was a" (a bit naughty, but never mind), we have:

$$p(l'_{i,j,t}|a) = \sum_{i'}\sum_{j'}\sum_{t'} p(l_{i,j,t}|a \wedge l_{i',j',t'})p(l_{i',j',t'}). \tag{4}$$

   For this exercise, assume that actions are deterministic. (Press the linebreak `Toggle Prob. Actions` if necessary so that a string to this effect is displayed in the `Robot Plan` window.) Recall that, with deterministic actions, if the robot is in pose (x, y, t), and performs action a, then the resulting pose is completely determined. Under this assumption, we can reproduce the effect of Equation (4) using the following steps. (i) Initialize a temporary matrix, say `workMatrix[][][]` to contain all zeros; this matrix will store the robot's new probability distribution. (ii) For each pose (i, j, t), get the *current* probability that the robot is in that pose (this is stored in `beliefMatrix[`i`][`j`][`t`]`), and simply add it to the cell `workMatrix[`i'`][`j'`][`t'`]`, where (i', j', t') is the pose that would result from performing action a in pose (i, j, t). (iii) When this is finished, `workMatrix[][][]` will store the updated probabilities given by Equation (4); so it can be copied into `beliefMatrix[][][]`. Here is the pseudo-code.

```
Initialize workMatrix[][][] to be everywhere zero.
for all $i$ , $j$ , $t$ in the standard range
    Compute the pose $(i', j', t')$ that results from
        performing action $a$ in pose $(i, j, t)$
    add beliefMatrix[$i$][$j$][$t$] to workMatrix[$i'$][$j'$][$t'$]
end for all
Copy workMatrix[][][] to beliefMatrix[][][].
```

You should convince your self that this algorithm will duplicate the effect of Equation (4). *Make sure you do convince yourself of this: you will need to understand how this works for the next exercise!*

To compute the effect of action a in pose (i, j, t), use the `WorldMap` method

`fillPoseOnAction(Pose` p`,int` i`, int` j`, int` t`, Action` a`).`

This method is a little tricky to use: the `Pose` p represents the *new* pose that will result after action a is performed in pose (i, j, t). You have to create an object in the class `Pose` to hold this result, and then execute the method, thus:

```
Pose tempPose= new Pose();    // Work variable
...
map.fillPoseOnAction(tempPose,i,j,t,a);
```

You can then look at the member variables of `tempPose` to get the new position and orientation resulting from performing the action. Don't create a new `Pose`-object every time you go through the loop.

4. (4 marks) Now modify the version of `updateProbabilityOnAction` that you wrote in the previous exercise, so that it allows for actions to be probabilistic. To determine *whether* actions are probabilistic, you need to execute the `probActions()` method in the value of the `probActionToggler` like this:

   `probActionToggler.probActions();`

   The value of `probActionToggler` is an the instance of the class `ProbActionToggler`, and the method `probActions()` returns `true` if actions are probabilistic, and `false` otherwise. So your code for `updateProbabilityOnAction(Action` a`)` will have the structure

```
if (!probActionToggler.probActions()){

    whatever your code was for the previous exercise
    }
else{

    the guts of your code for this exercise.

    }
```

The basic idea is simple enough. You want to compute $p(l'_{i,j,t}|a)$, where a is some action that the robot *tried* to perform. Equation (4) is still valid in this case. The snag is just that you cannot compute $p(l_{i,j,t}|a \wedge l_{i',j',t'})$ quite so easily as before.

However, help is at hand! Fix the intended action a. Now suppose that the parameter of this action, having been subject to random variation, takes the value u. We may safely assume that u is any number between 0 and 20, because this accounts for all of the probability mass in the simulator's model of action noise. And let us write $\mathbf{a}_u$ (in boldface) for the proposition that the robot tried to perform the action a, but that the parameter of this action actually took the value u.

Probability theory tells us that

$$p(l_{i,j,t}|a \wedge l_{i',j',t'}) = \sum_{u=0}^{u=20} p(l_{i,j,t}|\mathbf{a}_u \wedge a \wedge l_{i',j',t'})p(\mathbf{a}_u|a \wedge l_{i',j',t'}).$$

Moreover, it is reasonable to make the following independence assumptions:

$$\begin{aligned} p(l_{i,j,t}|\mathbf{a}_u \wedge a \wedge l_{i',j',t'}) &= p(l_{i,j,t}|\mathbf{a}_u \wedge l_{i',j',t'}) \\ p(\mathbf{a}_u|a \wedge l_{i',j',t'}) &= p(\mathbf{a}_u|a). \end{aligned}$$

Now, for each u ($0 \leq u \leq 20$), we can compute $p(l_{i,j,t}|\mathbf{a}_u \wedge l_{i',j',t'})$, using linebreak `map.fillPoseOnAction`, exactly as in the deterministic case. (Just make sure you call it with an action having parameter u.) So all that remains is to compute the numbers $p(\mathbf{a}_u|a)$. But we are assuming that the parameter u is sampled from a normal distribution with mean 10 and standard deviation 3.33. And you have code provided to compute this. The static function

```
public static double probabilify(int correctValue,
                                 int actualValue)
```

in the class `WorldMap` returns the probability that sampling a normal distribution with mean *correctValue* and standard deviation 1/3 of *correctValue* will yield *actualValue*. So a call of the form

```
WorldMap.probabilify(10, u);
```

should do the trick. Otherwise, you are on your own.

Fig. 2 shows what the working program should look like.

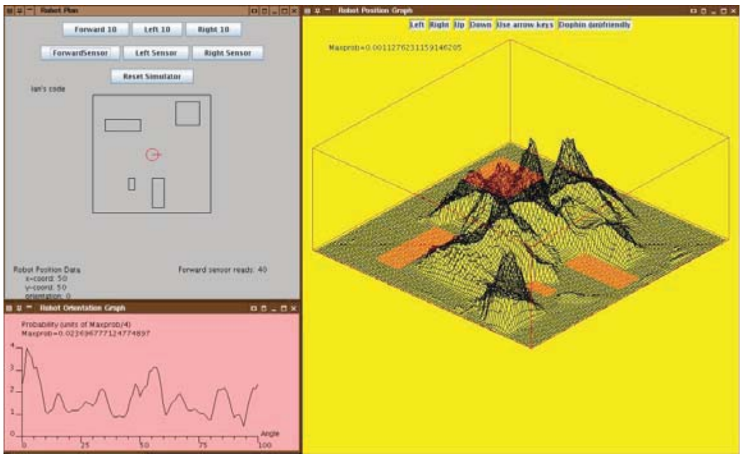


Figure 2: View of the working program

5 Completing your work

As soon as you have completed your work, you must run the program `submit` while you are in your `HOME/COMP14112/ex1` directory. *Do not wait until it is about to be marked, or ARCADE will probably think you completed late!* `submit` will only work if you have named your files correctly - check for minor filename variations such as the wrong case of character. The required files are named as follows.

`robot/RobotBeliefState.java`.

Then, as soon as you are next present in the School (e.g. straight away) run `labprint` and go immediately to the printer to collect your listing otherwise someone else could steal your work, and you could get accused of copying from them. When you get marked, the demonstrator will write comments on the listing. *You must keep the listing in your portfolio folder afterwards.*

For the face-to-face feedback and marking process, you should do the following when your demonstrator arrives.

1. Run `submit-diff` to show there are no differences in your work since you submitted it.
2. Have the feedback and marking process.
3. Run `submit-mark` to submit your mark. This will require the mark and a marking token, given to you by the demonstrator.

For each exercise, you will receive up to 80/working code it which you can explain to the demonstrator. You will receive up to 20/clear, well-structured, appropriately commented code.

COMP14112

Artificial Intelligence Fundamentals

Examples Classes - Part One

Second Semester
2013-2014

School of Computer Science
University of Manchester

COMP14112 Artificial Intelligence Fundamentals
Examples Class 1

Xiao-Jun Zeng
School of Computer Science,
University of Manchester,
Manchester M13 9PL, U.K.

1. What does it mean to say that an agent's subjective probability (degree of belief) in an event is 0.75?

2. What are your subjective degrees of belief in the following events?
   a) I will win the national lottery jackpot next week. (Assume you will buy one ticket.)
   b) The Earth will be destroyed by an asteroid some time next year.
   c) The number 585450317 is prime.

3. In the context of probability theory, explain the terms
   a) mutually exclusive;
   b) jointly exhaustive;
   c) partition.

   Given an example of how partitions naturally arise in the context of robot localization.

4. Let a single robot be a point object locating at some position in a square arena with 10×10 square grid.
   a) Assume that there is no obstacle in the arena and the robot is equally likely locating at each position. What is the probability that the robot is located at each position?
   b) Assume that the robot is equally likely locating at each position but there are two obstacles in the arena, one occupies 3x3=9 positions and the other 4x4=16 positions. If the robot can not locate at those positions occupied by obstacles, then what is the probability that the robot is located at each position?
   c) Under the same condition as b), if we assume further that the robot has 10 possible orientations, then what is the probability that the robot is located at each pose, where the pose is defined as the triple (i, j, t) of position and orientation together.

5. Let p be a probability distribution on Ω. Let E and F be events in Ω. Prove that
   a) $p(E^C) = 1 - p(E)$
   b) $p(E \vee F) = p(E) + p(F) - p(E \wedge F)$

   (Hint: You need to revise some basic properties about sets in order to complete the above proof).

COMP14112 Artificial Intelligence Fundamentals
Examples Class 2

Xiao-Jun Zeng
School of Computer Science,
University of Manchester,
Manchester M13 9PL, U.K.

In the sequel, p is a probability distribution on sample space Ω, and E and F be events in Ω. All derivations should employ only the axioms (K1) and (K2) of probability theory, or theorems derived from these axioms in the lecture notes.

1. List four different types of sensors used on mobile robots. Briefly describe their characteristics.

2. A mobile robot represents its degrees of belief about its current pose in terms of a probability distribution. Explain how these degrees of belief should be modified on the basis of sensor information. Why is Bayes' theorem useful here?

3. Let E and F be events with $p(E) > 0$. Prove that
   a) If $p(F) = 0$, then $p(F \mid E) = 0$
   b) If $E \subseteq F$, then $p(F \mid E) = 1$

4. Let $E_1,..., E_n$ also be events and form a partition, and $p(E \wedge E_1),..., p(E \wedge E_n)$ be all positive. Prove that
$$p(F \mid E) = p(F \mid E \wedge E_1)\, p(E_1 \mid E) + ... + p(F \mid E \wedge E_n)\, p(E_n \mid E)$$

5. Let $E_1,..., E_n$ be events and form a partition, and $p(E), p(E_1),..., p(E_n)$ be all positive. Prove that
$$p(E_i \mid E) = \frac{p(E \mid E_i)\, p(E_i)}{p(E \mid E_1)\, p(E_1) + ... + p(E \mid E_n)\, p(E_n)}$$

6. A scientist conducts an experiment to test three hypotheses *h1*, *h2* and *h3*. He knows that exactly one of these hypotheses must be true. Let *t* represent the event that the outcome of a certain experiment is positive. On the basis of theoretical calculations and previous experiment, the scientist assigns the following degrees of belief:

   p(h1) = 0.4
   p(h2) = 0.4
   p(h3) = 0.2
   p(t|h1) = 0.1
   p(t|h2) = 0.5

$$p(t|h3) = 0.99$$

Suppose the scientist performs the experiment and indeed obtains a positive outcome. If this is all he comes to know, what degrees of belief should he then assign to the three hypotheses?

COMP14112 Artificial Intelligence Fundamentals
Examples Class 3

Xiao-Jun Zeng
School of Computer Science,
University of Manchester,
Manchester M13 9PL, U.K.

The Multi-Door and Multi-Stage Monty Hall Problem

In lecture 3, we discuss the three-door Monty Hall problem. That is, the game is played in two stages — Stage 1: Choose a door; Stage 2: Switch or stick with your choice. Here, the probability that you win the car is 2/3 if you switch your choice. So, yes. It *is* to your advantage to switch.

Let us now generalize this game to N doors. That means, the game is played in (N - 1) stages.

Stage 1: Choose a door

For stages X = 2 through (N – 1):

Monty Hall shows you one of the "goat doors". Then, you either switch to one of the remaining (N - X) doors or stick with your choice.

1. Consider N=*4*. You get to choose a door at the first stage, and for the next two stages, you can either switch or stick. So the various permutations are:

   (1) Choose, Stick, Stick
   (2) Choose, Stick, Switch
   (3) Choose, Switch, Stick
   (4) Choose, Switch, Switch

   Questions:

   1) Which of these four strategies has the highest probability of winning in this scenario?
   2) Why is this best strategy?
   3) What is the probability of winning with this strategy?

   (Hint: The same as the three-door problem, you can use the conditional probabilities and Bayes' Theorem by calculating the probability of wining for every strategy or finding out the strategy with the highest probability of winning. Alternatively, if you don't like the solution based on probability theorem, you can develop a computer simulation to find out the solution, for example, by extending and modifying the C# resource codes for three-door problem given in the link below: http://www.codeproject.com/KB/cs/montyhall.aspx)

2. Now consider the general n-door problem and answer the following questions*:

   1) What strategy has the highest probability of winning in this scenario?
   2) Why is this best strategy?
   3) What is the probability of winning with this strategy?

   (*These are challenging questions and you do not have to find out the answers but you are encouraged to think about these questions)

COMP14112 Artificial Intelligence Fundamentals
Examples Class 4

Xiao-Jun Zeng
School of Computer Science,
University of Manchester,
Manchester M13 9PL, U.K.

Using the library catalogue, retrieve a copy of the article

A.M. Turing, “ComputingMachinery and Intelligence”, Mind 59(236): 433–460, 1950.

1. Briefly describe the ‘Imitation Game’, as set out in Turing’s article.

2. On which page does Turing refer to the possibility of human cloning (though not by that name)?

3. Whom does Turing credit with inventing the idea of a digital computer, and when?

4. What, according to Turing, does it mean to say that digital computers are universal machines?

5. By what date did Turing think it would be possible to program a computer to play the imitation game so that an average interrogator would not have more than 70% chance of making the right identification after five minutes of questioning?

6. Turing considered various objections to his view. Choose the two best such objections (in your opinion), briefly describe them, and say whether you think they are valid.

7. Do the same for the two worst objections: say why you think they are bad objections.

8. Retrieve the paper

   Rodney A. Brooks, “Elephants Don't Play Chess”, Robotics and Autonomous Systems, Vol. 6, No. 1-2, Pages 3-15, 1990

   Read the paper and compare with Turing’s paper by listing 3-5 main different opinions about AI and discussing your view about these opinions.