Reasoning about uncertainty

Rule-based systems are an attempt to embody the knowledge of a human expert within a computer system.

Human knowledge is often imperfect.

- it may be incomplete (missing facts)
- or inexact (rough approximation rather than exact value)
- or uncertain (we think something *might* be true, but are not sure)
- or even inconsistent!

To construct an RBS that recreates human reasoning, we must find ways of reasoning in the presence of uncertainty.

 CSCU9T6 1

Kinds of Uncertainty

Generally, uncertainty in AI refers to situations where:

- information is incomplete, or
- information is not reliable, or
- language used is inherently imprecise, or
- there is conflicting information, or
- information is approximate

Many approaches have been developed to deal with this in AI systems. We shall look at these approaches:

- Reasoning with uncertain in Rule-Based Systems
  - Certainty factors, and probability – this lecture
- Fuzzy sets and fuzzy logic – next lecture

 CSCU9T6 2

Dealing with uncertainty

The standard approach of a rule-based system is:

Ask the user for information, e.g. `Yes/No` or menu choice, then proceed with further steps, based on the answer.

Handling uncertainty means accommodating responses like:

- 'maybe'
- 'probably'
- 'don't know'
- 'well, it's somewhere between `medium` and `high`'

The information which we have may be uncertain. But we may still be able to draw conclusions, perhaps with some level of uncertainty attached.

Uncertainty may also apply to the reasoning process:

```
If p and q then it is likely that . . .
```

 CSCU9T6 3

Certainty Factors

This was the method used by MYCIN, one of the early trailblazing rule based systems, and since then adopted by many other systems, including e2glite.

A *certainty factor* (CF, and also called *confidence factor*) involves a measure which represents a level of confidence that some condition is true.

Certainty factors have been implemented differently in different systems.

In e2glite, certainty factors are expressed as percentages. A CF of 100% means complete certainty, and a CF of 0% means no certainty.

(In MYCIN, certainty factors were implemented as numbers from -1 to 1).

 CSCU9T6 4

Certainty Factors in e2glite

In operation the user may be asked questions in the form

Do the headlights dim when you try the starter?:
o yes
o no
o I don't know

How confident do you feel about your response?

Very uncertain (50%) o o o o o o Very certain (100%)

The user must choose a response (yes, no, or I don't know) and choose a level of confidence (from 50 – 100%) in that response. If the user's confidence level in a response option is below 50%, then a different response option should be chosen.

 CSCU9T6 5

Certainty Factors in Prompts

The knowledge base contains the following prompt:

```
PROMPT [the headlights dim] YesNo CF
"Do the headlights dim when you try the starter?"
```

This is a `YesNo` prompt. When it is "run", it will attempt to determine a boolean value for the attribute `[the headlights dim]`.

Example: If the user replies "Yes" and chooses confidence level 50%, then `[the headlights dim]` will get the value `true`, with CF 50%.

Certainty factors may be used with all kinds of prompts, simply by adding `CF` after the prompt type.

 CSCU9T6 6

Certainty Factors in Rules

A rule based system can propagate CFs via rules to give CFs to conclusions. For this, rules too may have CFs associated with them. E.g.

```
RULE [Is the petrol tank empty?]

If [the result of trying the starter] = "the car cranks normally" and

[a petrol smell] = "not present when trying the starter"

Then [the petrol tank] = "empty" @ 90
```

This says that if we know that the two conditions in the premise hold, then this knowledge gives a *degree of belief* that `[the petrol tank]` has the value `"empty"`.

If no CF is given for a rule it is assumed to be 100%.

 CSCU9T6 7

An abstract representation

Attributes:

A = [the result of switching on the headlights]

B = [the result of trying the starter]

C = [the petrol tank]

D = [a petrol smell]

= [the headlights dim when trying the starter]

F = [the amount you are willing to spend on repairs]

GOAL = [the recommended action]

 CSCU9T6 8

An abstract representation (2)

Values:

a1 = "nothing happens"
a2 = "they light up"
b1 = "nothing happens"
b2 = "the engine tumbles normally"
b3 = "the engine tumbles slowly"
c1 = "empty"
c2 = "not empty"
d1 = "present when trying the starter"
d2 = "not present when trying the starter"
e1 = true
e2 = false
f1 = a number in [0, 500]
goal1 = "recharge or replace the battery"
goal2 = "refuel the car"
goal3 = "recharge or replace the battery"
goal4 = "wait 10 minutes, then restart flooded car"

 CSCU9T6 9

An abstract representation (3)

Rules:

R1 = [Is the battery dead?]
$((A = a1) \vee (B = b1)) \rightarrow (GOAL = goal1)$

R2 = [Is the car out of petrol?]
$(C = c1) \rightarrow (GOAL = goal2)$

R3 = [Is the battery weak?]
$((B = b2) \vee (B = b3)) \wedge (E = e1) \wedge (F > 24.99)) \rightarrow (GOAL = goal3)$

R4 = [Is the car flooded?]
$((B = b2) \wedge (D = d1)) \rightarrow (GOAL = goal4)$

R5 = [Is the petrol tank empty?]
$((B = b2) \wedge (D = d2)) \rightarrow (C = c1)$

 CSCU9T6 10

Expression in propositional logic

A1 means (A = a1), B1 means (B = b1), B2 means (B = b2),
B3 means (B = b1), C1 means (C = c1), D1 means (D = d1),
D2 means (D = d2), E1 means (E = e1), F1 means (F > 24.99),
G1 means (G1 = goal1), G2 means (G2 = goal2),
G3 means (G3 = goal3), G4 means (G4 = goal4)

Rules:

R1: $(A1 \vee B1) \rightarrow G1$
R2: $C1 \rightarrow G2$
R3: $(B2 \vee B3) \wedge E1 \wedge F1 \rightarrow G3$
R4: $(B2 \wedge D1) \rightarrow G4$
R5: $(B2 \wedge D2) \rightarrow C1$

Combining Certainty Factors

When conditions are combined using **and** or **or**, the following rules are used to calculate the CF of the combination:

CF(**A and B**) = minimum of CF(**A**), CF(**B**)

CF(**A or B**) = maximum of CF(**A**), CF(**B**)

When applying a rule of the form **If P Then Q @ n**, the CF of the conclusion **Q** is determined by combining the CF of the premise **P** and the CF of the rule itself, **n**:

CF(**Q**) = CF(**P**) × **n**/100

(Division by 100 is needed to get a result in the range 0-100.)

Confidence Factors (Example)

Here is a set of rules:

```
If B or C Then A @ 60
If D and E and F Then B @ 100
If G or H Then C @ 75
```

Suppose that we know CFs for D, E, F, G and H as follows:

```
CF(D) = 80%          CF(E) = 50%
CF(F) = 90%          CF(G) = 20%
CF(H) = 80%
```

Find the CF for A.

Confidence Factors (Example)

Derivation:

```
CF(C)  =  CF(G or H) X 75%
       =  max(20%,80%) X 75%
       =  80% X 75%
       =  60%

CF(B)  =  CF(D and E and F) X 100%
       =  min(80%,50%,90%) X 100%
       =  50%

CF(A)  =  CF(B or C) X 60%
       =  max(50%,60%) X 60%
       =  60% X 60%
       =  36%
```

Combining Rules

It may happen that a conclusion is supported by more than one rule. For example, in addition to the rules above, we might have another rule for **A**:

```
If P and Q Then A @ 80
```

This could lead to a quite independent calculation of a CF for **A**. How should this be combined with the other one to obtain an overall CF?

Different solutions have been adopted in different rule-based systems. The next slide shows the solution used in e2glite…

 CSCU9T6 15

Combining Rules (2)

Suppose that we can find certainty factors CF1 and CF2 for a conclusion **C** by application of two independent rules. We can then calculate an overall CF by applying the following formula:

$$CF(C) = CF1 + CF2(100 - CF1)/100$$

- Combined confidence in C
- Influence of confidence due to the first rule
- Influence of confidence due to the second rule, reduced by remaining uncertainty about the first rule

 CSCU9T6 16

Combining Rules (Example)

In the example we saw that the first rule for **A** gave a CF of 36% for **A**.

Suppose that the second rule for **A** gave a CF of 75% for **A**.

Then the formula would be used to give a final CF for **A**:

$$CF(\mathbf{A}) = CF1 + CF2(100 - CF1)/100$$

$$CF(A) = 36 + 75(100 - 36)/100$$
$$= 84$$

 CSCU9T6 17

Confidence Factors - Evaluation

Although they resemble probabilities, confidence factors are not the same. Unlike probabilities, there is no rigorous mathematical basis for using CFs. The formulas used are somewhat arbitrary.

The use of CFs has therefore been criticised (with some justification) for not being rigorously founded.

It is also only as good as its input (users' guesses).

But it has been found to be remarkably resilient. Systems which use them seem to work reasonably well in practice, in comparison with human experts.

Anyway, (as far as we know) human experts don't use rigorous mathematical theory either.

 CSCU9T6 18

Certainty factors in MYCIN

Certainty factors were invented by Shortliffe and Buchanan for use in the MYCIN rule based system.

They were a response to the perceived failures of systems based on probability theory (Bayes theorem).

Mycin uses certainty factors from -1 to 1

- A CF of -1 represent total disbelief in a hypothesis or rule
- A CF of 0 represents no belief or disbelief ("don't know")
- A CF of 1 represents total belief

When combining two CFs, the rule used depends on if they are both positive, both negative, or mixed.

e2glite does not handle reasoning about disbelief, so uses only positive CFs. The underlying approach, however, is the same as in MYCIN.

 CSCU9T6 19

Bayesian methods: the past (and the future?)

Older rule-based systems used Bayesian probability to handle uncertainty.

These simple-Bayes models rely on assumptions that:

1) Hypotheses are mutually exclusive and exhaustive.

2) Pieces of evidence were conditionally independent (for a given hypothesis)

In practice, these assumptions are often inaccurate. For example: Hypothesis 1: "The car is out of petrol" and Hypothesis 2:"The battery is dead" are not mutually exclusive.

Failures of the simple-Bayes model led researchers to invent certainty factors. However, more sophisticated Bayesian models - belief networks - avoid these problems and are increasing in popularity.

20

Further reading:

From Certainty Factors to Belief Networks, Heckerman and Shortliffe, 1992

http://research.microsoft.com/en-us/um/people/heckerman/HS91aim.pdf

Also, the entry under "Expert Systems" in wikipedia.

