Module 4: Fuzzy Theory, Reasoning, and Rules

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

Module 4 Study Guide and Deliverables

Module Theme: Fuzzy Theory, Reasoning, and Rules

Readings:	• Module 4 online content
Assignments:	• Lab 4 due Sunday, October 3, at 6:00 AM ET • Assignment 4, due Wednesday, October 6, at 6:00 AM ET
Live Classrooms:	• Wednesday, September 29, from 8:00 PM to 9:00 PM ET • Thursday, September 30, from 8:00 PM to 9:00 PM ET • Live Office: Wednesday and Thursday after Live Classroom, for as long as there are questions

Rule-based Systems

Learning Objectives

Despite their success, technologies like neural nets and SVM's are knowledge free—"dumb" in many ways. To illustrate this, suppose that your child answered "105" when asked "how much is 98 + 9?" But when you ask "why", he answers "… *because I just happen to know that 97 + 9 = 106 as well as 99 + 9 = 108, and 98 is evenly between 97 and 99, so the answer should be evenly between 106 and 108*." In other words, they are entirely data-driven, and unaware of the rules of addition. Even in the non-mathematical world, we strive for understanding. For example, we look to rules of history when engaging in diplomacy.

In this section, we will learn about rule-based systems but we will also need to discuss the issue of where the knowledge base comes from. Sources for ***expert systems***—a type of rule-based systems—have been mainly human. This may appear to be inconsistent with machine learning. However, for many applications such as medicine, in order for machine-learned artifacts to be acceptable to society they must be explainable. For this reason, there are strong ties between machine learning and knowledge-based systems.

Loading [Contrib]/a11y/accessibility-menu.js

is part of the module, you will be able to do the following:

1. Design rule-based systems.
2. Collect knowledge.

Definition

Rule-based System

- Knowledge can be represented as rules.
- Knowledge separated from inference process

Additional properties:

The great advantage of rule-based systems is that they can explain their reasoning.

There is little limit on how deep reasoning can go, though rule-based systems in practice have seldom been found to go very deep.

Typical Origin of Rule-based System

The following example shows how an expert rule-based system can come about. There is typically an individual whose knowledge and expertise are considered valuable, and a desire to codify it.

Give users a step-by-step method of analyzing their physical fitness, offering techniques and strategies which lead to enhanced fitness practices and habits.

Allow long-term use of the tool to track progress and to adapt the system's advice.

Alternatives to Rule-based Systems

Rule-based systems are used when traditional processing, simulation etc. are not practical, but expertise is available.

- Straight processing
  - hard to adapt to new expertise, must build reasoning into every new part
  - long time to prototype
- Model and simulate
  - insufficient understanding of the process
- Spreadsheet/existing tool

Loading [Contrib]/a11y/accessibility-menu.js

- lacks symbolic processing
- Unexplained deep learning

Knowledge for Rule-based Systems

In this section, we will discuss a common structure of ***knowledge***.

Data/Facts

The most elementary form of knowledge is data. You can think of this as an entry in a table—more specifically, as the value of a field in a record. For example, if each record corresponds to a name, and there is a *type* field, then the *Antonio* record has “animal” for the value of its type. The record has “stripes” in its characteristic field.

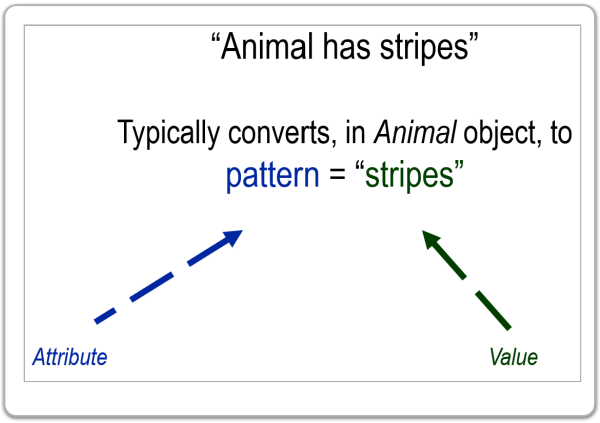
In the context of rule-based systems, we use the word *fact* for data that’s provided in context—in other words “Antonio has stripes,” not just “stripes” or “24.”

> Fact: Data in context, e.g.,
>
> - Fact: Antonio is an animal.
> - Fact: Antonio has stripes.

Facts are generally of the form

field (or variable, typically attribute of an object) has value ….

We have already seen “facts” in the context of propositional logic.



What is Knowledge?

Rule-based systems are based on **knowledge** (and are more generally referred to as ***knowledge-base systems***).

Loading [Contrib]/a11y/accessibility-menu.js

1. ***Factual knowledge***. Knowledge includes facts, but is more general. Here, we give examples.
    - e.g., “there were two snowstorms in MA between August 1994 and December 1994”
2. Non-data ***declarative knowledge*** consists of statements that (depending on the circumstances) can be either true or false. In other words, they are booleans.
    - e.g. “Liberal kings are rare”

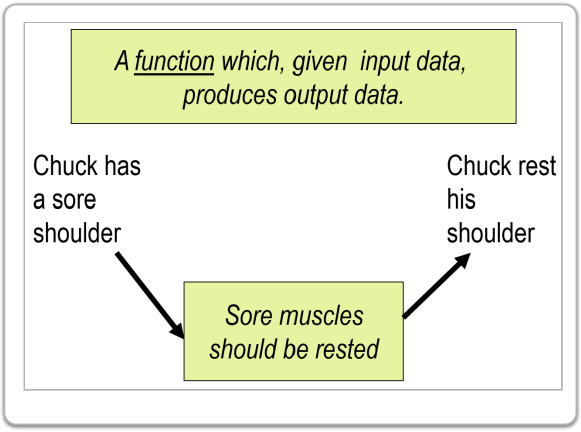
Rule-based systems usually deal with factual and declarative knowledge, although there are systems that deal with **procedural knowledge** (how to do something) and **meta-knowledge** (knowledge about knowledge).

3. Non-data ***procedural knowledge***
    - e.g. “To replace a bike chain, first …”
4. ***meta-knowledge***
    - knowledge about knowledge

Definition of Non-fact Knowledge

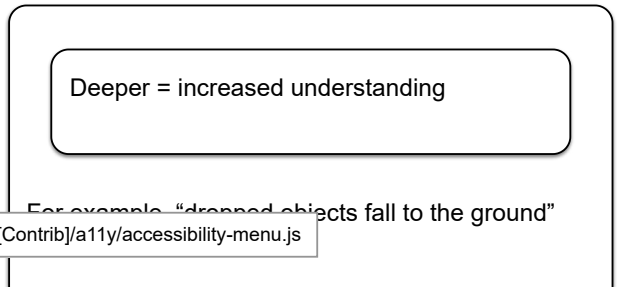
A unit of knowledge that isn’t a fact is a kind of **generator**: given fact(s), it may produce other fact(s), as in the figure.

We have already seen knowledge in the form of first-order logic.



Shallow vs. Deep Knowledge

We generally think of knowledge as having **depth**, as in the example. There has always been an attempt to drive rule-based system knowledge deeper, though most rule-based systems I have encountered do not reason as deeply as they could in theory.

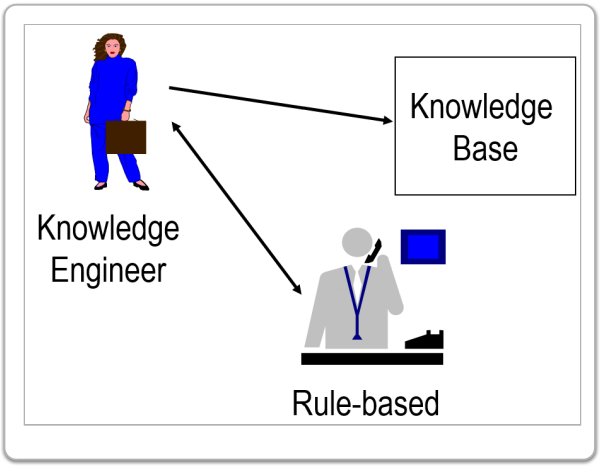


Loading [Contrib]/a11y/accessibility-menu.js

- deeper: “any two bodies in space attract each other”
- deeper: “any pair of entities exist in a space-time curve”

Architecture

In this section we describe the components of a rule-based system and how they relate.



Traditionally, expert system knowledge has been encoded in a pre-arranged format by a knowledge engineer interviewing the rule-based. A knowledge engineer is a little like a business analyst, but more technical because they encode the knowledge.

There is usually a single rule-based. When there are more than one, they typically do not agree on everything, which leaves open the question of who selects.

Problems to be Handled

- Separate knowledge from reasoning
- Represent knowledge
- Decide reasoning method
- Integrate user interface

The most important feature of expert systems is that they confine the knowledge and the reasoning mechanism to separate modules. The question of how, exactly, to represent (express) the knowledge has also to be decided. There is more than one way to perform reasoning.

Separating Knowledge & Inference

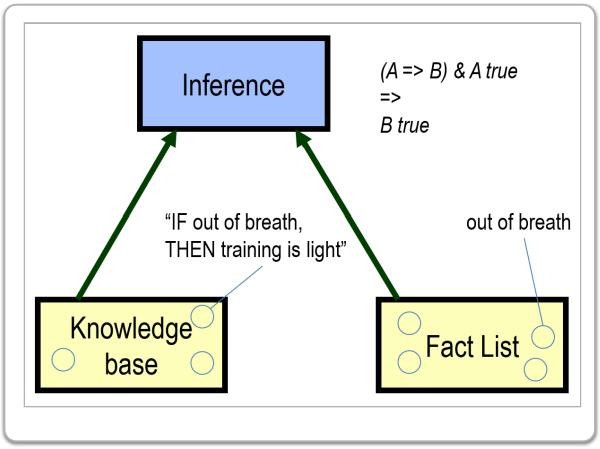
Loading [Contrib]/a11y/accessibility-menu.js

The inference mechanism, known as the ***inference engine***, performs logic such as ***modus ponens***. Modus ponens is the following logical operation:

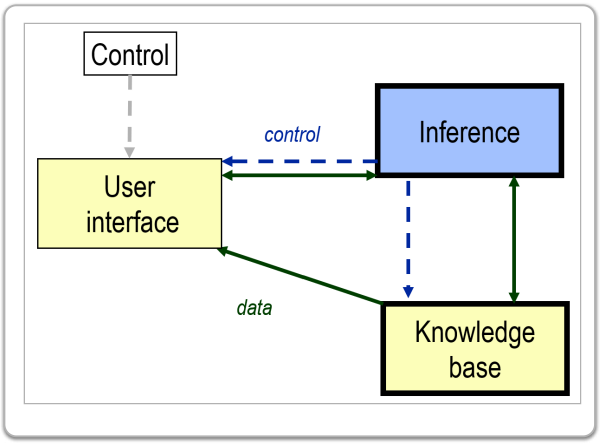
```
If it is known that
  statement A implies statement B,
  and also that statement A is true,
then it follows that
  statement B must be true.
```

The most common way to represent knowledge is to express it in multiple "*If … then…*" rules, as shown in the example. We shall describe such rules in detail.

The inference engine infers new facts from the knowledge base and the fact list. For example, the inference engine uses modus ponens to infer the fact "training is light" from the rule "IF out of breath, THEN training is light" and the fact "out of breath." The fact "training is light" is then added to the fact list.



The figure below shows the three principal components of expert systems. Date flows are shown with solid lines: data flows everywhere except from the inference engine. On the other hand, the inference engine tends to be a center of control (expect for overall control of the expert system).



Expert System Architectures

Loading [Contrib]/a11y/accessibility-menu.js

We will discuss the expert system types listed in the figure. The most important one is **rule-based**. We discuss **model-based** below. **Case-based** is used to solve problems for which there is an existing set of problem-solution pairs (typically mush more complex than simple input-output pairs for neural nets).

1. Rule-based
2. Model-based
3. Case-based
4. Other

Rule Terminology

Rules are the basic form for many knowledge bases. A rule is composed of a conjunction (`AND`-ing) of facts in the `IF` part and a disjunction (`OR`-ing) of facts in the THEN part. This form can't be decomposed into simpler rules. (An `OR`-ing of facts in the `IF` part and an `AND`-ing of facts in the `THEN` part can be decomposed).

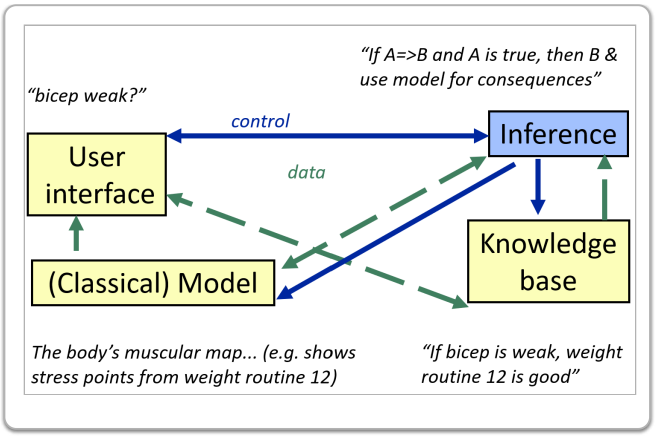
```
IF
   animal has stripes           ← antecedent (fact)

   AND                          ←conjunction

   animal roams in large herds  ← antecedent

THEN
   animal is zebra              ← consequent

   OR                           ← disjuction

   animal is okapi              ← consequent
```

Model-based Expert Systems

Pure expert systems are based on knowledge about the problem (the knowledge base, in one module) and a means for reasoning about it (inference, in another module). This is different from most programs, which are built around ***models***. A model is a conception that's either described in writing or else in the heads of developers. For example, a train-scheduling program is based on a conception of trains and stations and passengers taking trips.

A ***model-based expert system*** combines both. The design depends on the problem, but the inference engine calls on the model as well as the knowledge base.

Loading [Contrib]/a11y/accessibility-menu.js

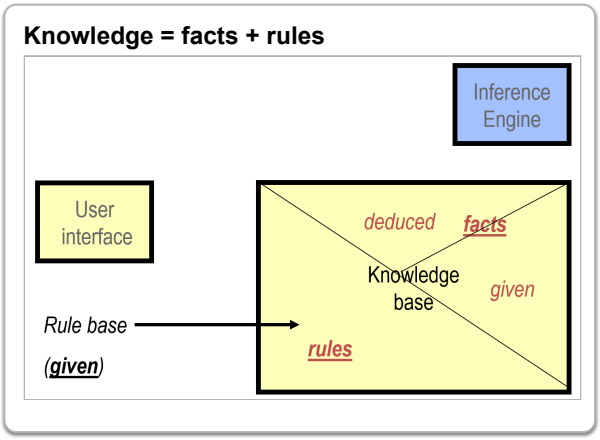


Inference Methods

There are two principal kinds of rule-based system reasoning.

1. In ***forward chaining***, the known facts are used as the starting point.
2. In ***backward chaining***, the fact we want to know the truth of (there must be one) is the starting point.

The collection of rules (the **rule base**) for a rule-based system is given, and typically does not change. Some facts are given—when the system begins or else supplied by the user at runtime in response to prompts from the system. Additional facts may be deduced at runtime, as in the example above. In a sense, the latter are learned.



Forward Chaining

In this example, there are two rules and a single known fact.

Initial fact list: *Brian is a manager*

Rules:

1. `IF` *`X has a large office`* `THEN` *`X is happy`*

Loading [Contrib]/a11y/accessibility-menu.js

nager `THEN` *`X has a large office`*

Goal: Generate all consequents

Mechanics of Forward Chaining

> Apply all rules repeatedly, adding to fact list each time, until no new fact are found.

Initial fact list: *Brian is a manager*

After first application of all rules:

1. Brian is a manager
2. Brian has a large office

After second application of all rules:

1. Brian is a manager
2. Brian has a large office
3. Brian is happy

Forward chaining cycles through the two rules. On the first cycle, the additional fact *Brian has a large office* is deduced because Rule 2 “fires.” The two rules are tried again, and this time Rule 1 fires, deducing *Brian is happy*.

Rather than generate every deducible fact, one sometimes has a target fact such as *Brian has a large office*. The process stops when that target fact is deduced.

Backward Chaining

Backward chaining requires a target fact such as *Brian is happy*. We look at every rule containing this fact as consequent. This is only rule 1 in the example (in general there could be several such rules). Now we try (recursively) to prove every antecedent—they are the new target facts.

Initial fact list: *Brian is a manager*

Rules:

1. `IF X has a large office THEN X is happy`
2. `IF X is a manager THEN X has a large office`

Goal: Answer query: *Is Brian happy?*

THe figure below shows the backward chain of reasoning (steps i and ii).

Loading [Contrib]/a11y/accessibility-menu.js

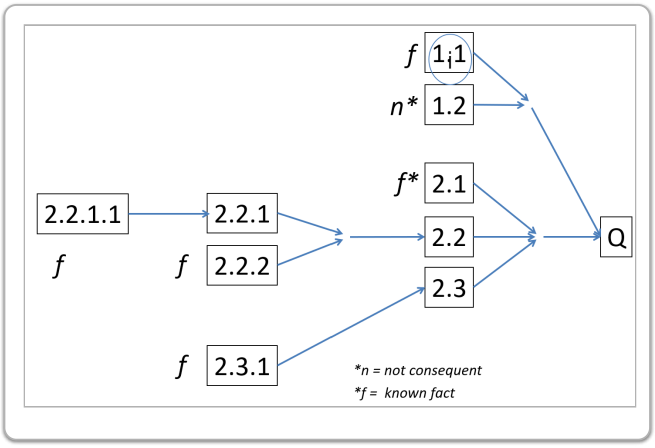
Brian is a manager \(\underrightarrow{\text{ii}}\) **Brian has a large office**

\(\underrightarrow{\text{i}}\) **Brian is happy**

Notes to Precede Backchaining Algorithm

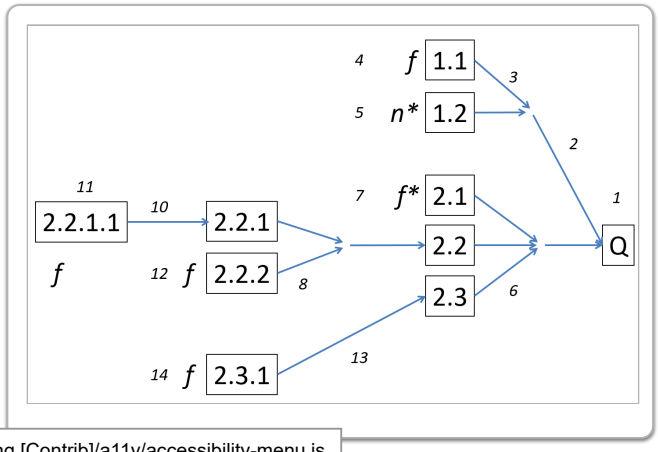
This schematic shows a set of five rules and a target fact Q.

There are two rules that infer Q. The first is of the form `IF 1.1 AND 1.2 THEN Q` (we are dispensing with full English statement of facts in order to emphasize the logic). We now try to establish fact 1.1. According to the figure, this is known. However, fact 1.2 is not known, nor is it a consequent (so there is no hope of proving it). For this reason, we abandon the rule `IF 1.1 AND 1.2 THEN Q` as a means for proving Q, and move on to seeing whether we can establish the antecedents of the rule `IF 2.1 AND 2.2 AND 2.3 THEN Q`.



This figure below shows the order in which facts are "visited." It turns out that the antecedents of the rule "`IF 2.1 AND 2.2 AND 2.3 THEN Q`" can indeed be established, and so Q has been established. Note that backward chaining does not add new facts to the fact list. Instead, it stacks recursive calls.

The appendix describes a Java implementation of backward chaining.

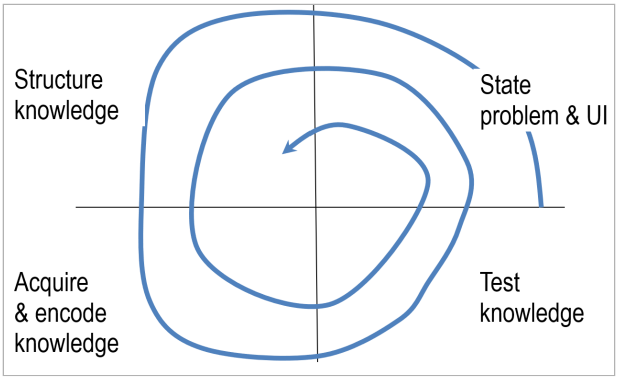


Loading [Contrib]/a11y/accessibility-menu.js

Knowledge Engineering

The process of knowledge acquisition is cyclic in nature, as in the figure, and necessarily agile. You typically start by specifying the problem, then structuring the knowledge, acquiring and encoding a batch, and then showing it to the expert.

Figure: Process of Soliciting Expert's knowledge

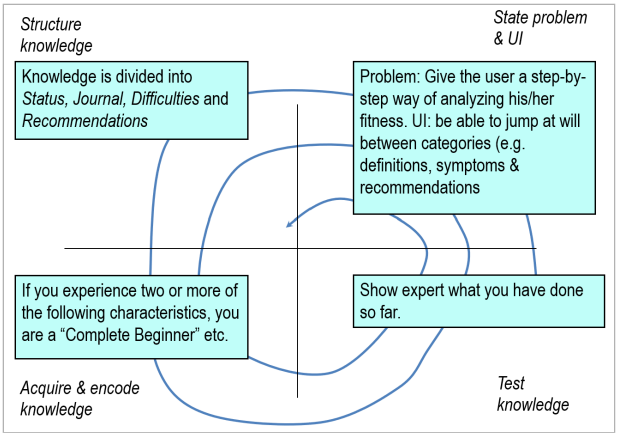


Examples

Fitness Expert System

This figure is an example that illustrates this process for a physical fitness advisor.

Figure: Process of Soliciting Expert's knowledge



Expert System Shell Example: pyCLIPS

Expert system shells facilitate the expression of facts and rules, and they control chaining. Python-base shells include PyKE and PyCLIPS. A simple PyCLIPS rule is shown above ("`IF quack THEN duck`"). PyClips allows for sophisticated facts, coded as objects.

Loading [Contrib]/a11y/accessibility-menu.js

Other systems use XML.

```
>>> import clips
>>> clips.Reset()
>>> clips.Assert("(duck)")
<Fact 'f-1': fact object at 0x00DE4AE0>
>>> clips.BuildRule("duck-rule", "(duck)",
"(assert (quack))", "the Duck Rule")
<Rule 'duck-rule': defrule object at 0x00DA7E00>
>>> clips.PrintRules()
MAIN:
duck-rule
```

Demonstration of CLIPS (optional video): A.I. - Artificial Intelligence, CLIPS Programming Language, quick run thru! Car Diagnosis! ACE

Diabetes Diagnosis Example: pyknow

Optional video: Expert system: Detecting Children Diabetes (please mute the sound when watching—the music is not related to content.)

Appendix: Expert System Example

Interview Radar Expert—Repair Expert System

There are two principal kinds of rule-based system reasoning.

The following is an edited transcript of an interview of a radar expert on problems that seamen may encounter. It illustrates the contrast between an expert's view of problems and their mode of expression with the desire of the knowledge engineer to encode it. This gap is bridged by skillful channeling.

> **Knowledge engineer:** The question is, supposing everything is up and running, and that same thing happened?
>
> **Expert:** Well, because of, let's say, and intermittent cable or a cable that is briefly grounded, suppose that the clock stops for a few milliseconds and then restarts. If it is literally a few milliseconds, like 3 or 4, or something like that, you may not notice; you wouldn't notice that symptom. For this to happen, and to be sure that you would recognize it, you would have to be there for, like 12 seconds.
>
> **Knowledge engineer:** To bring down the track module?

Loading [Contrib]/a11y/accessibility-menu.js

Knowledge engineer: What else could it mean?

Expert: Well, there, is the track – all right, there are some other cases. If you have an extremely large number of redundant tracks, one of the functions the track association module does is to monitor the track file for, you know, redundant tracks that have gotten into the system. You know, if two tracks crossing each other may become redundant, it's possible, as we have seen in the past few days, where if they turn off XXX in a severe environment, what tends to happen …

The knowledge engineer may draw an implication diagram as in the figure.

Turn into Diagram

```
Computer crashes within minutes of loading
          &
     Track association module error 17

               ↓

     Track association -----> Navigation
     module timeout           clock down
          &
     XXX turned off
          &
     Severe
     environment

               ↓

     Track association
     module overload ------> Turn on XXX & reload
```

The diagram is then converted to rules. This example used an expert system shell developed at Rutgers University but there are many shells, that make it relatively easy to encode once the knowledge is understood.

Turn into Rules

```
CCWSL & TAN17 --> TAMT

TAMT --> NCD

TAMT & XXXOFF & CENV --> TAMO

TAMO --> TOXXXR
```

Loading [Contrib]/a11y/accessibility-menu.js

Uncertainty and Bayesian Methods

Learning Objectives

After successfully completing this part of the module, you will be able to do the following:

1. Trace origin of Bayes' Law.
2. Compare `if … then` with Bayes' Rule.
3. Compute probabilities from prior probabilities.
4. Prune to obtain results.
5. Trade off uncertainty methods.
6. Apply to examples.

Uncertainty and Bayesian Networks

Sources of Uncertainty

There are at least two kinds of uncertainty within rules. The "facts" may not be solid, and even when they are, the rule as a whole may be not entirely valid.

$$\text{IF } \underbrace{ant1}_{\text{Fact uncertainty}} \underbrace{\text{ AND } ant2 \text{ THEN } cons1 \text{ OR } cons2}_{\text{Implication uncertainty}}$$

Symbolic Uncertainty

When people talk about uncertainty, they usually use descriptive language such as "more-or-less" and "sometimes." This is often the case when knowledge is obtained from experts. Data itself is seldom 100% consistent. If these uncertainty terms can be combined in an algorithmic manner as the experts do (e.g., what is the upshot of two antecedents that are "often" and "seldom" respectively), such an organization should be utilized.

Approach: use attributes with logical weight values.

Example: {`often`, `sometimes`, `seldom`}

$$\text{IF Smoker/sometimes AND Uneven/often}$$
$$\text{THEN Motivated/seldom}$$

Use when possible (e.g. combinatorially) … expert has handle on weighting

Loading [Contrib]/a11y/accessibility-menu.js

The table shows a way in which some uncertainly in rules and antecedents can produce consequents. For example, if we have the very sure rule \(A \text{ & } B =\gt{C}\), \(A\) and \(B\) are neither very sure nor very unsure (“in-between”), then \(C\) is also in-between.

Table: An Uncertainty Calculus For \(A \text{ & } B =\gt{C}\)

Rule	**\(A\)**	**\(B\)**	**\(C\)**
v. sure	v. sure	v. sure	v. sure
		in-between	in-between
		v. unsure	in-between
	in-between	in-between	in-between
		v. unsure	v. unsure
	v. unsure	v. unsure	v. unsure
in-between	…	…	…

Knowledge for Rule-based Systems

In this section, we will discuss a common structure of ***knowledge***.

Data/Facts

The most elementary form of knowledge is data. You can think of this as an entry in a table—more specifically, as the value of a field in a record. For example, if each record corresponds to a name, and there is a *type* field, then the *Antonio* record has “animal” for the value of its type. The record has “stripes” in its characteristic field.

In the context of rule-based systems, we use the word *fact* for data that’s provided in context—in other words “Antonio has stripes,” not just “stripes” or “24.”

Fact: Data in context, e.g.,

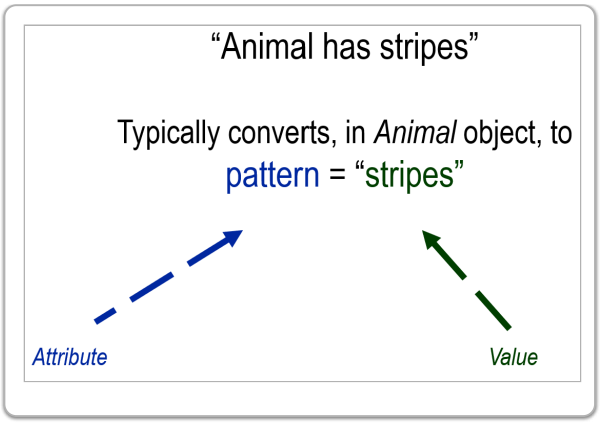
- Fact: Antonio is an animal.
- Fact: Antonio has stripes.

Facts are generally of the form

field (or variable, typically attribute of an object) has value ….

We have already seen “facts” in the context of propositional logic.

Loading [Contrib]/a11y/accessibility-menu.js



What is Knowledge?

Rule-based systems are based on **knowledge** (and are more generally referred to as ***knowledge-base systems***).

1. ***Factual knowledge***. Knowledge includes facts, but is more general. Here, we give examples.
    - e.g., "there were two snowstorms in MA between August 1994 and December 1994"
2. Non-data ***declarative knowledge*** consists of statements that (depending on the circumstances) can be either true or false. In other words, they are booleans.
    - e.g. "Liberal kings are rare"

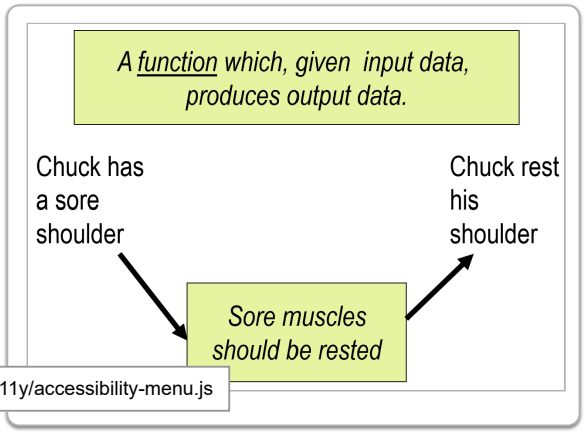
Rule-based systems usually deal with factual and declarative knowledge, although there are systems that deal with **procedural knowledge** (how to do something) and **meta-knowledge** (knowledge about knowledge).

3. Non-data ***procedural knowledge***
    - e.g. "To replace a bike chain, first …"
4. ***meta-knowledge***
    - knowledge about knowledge

Definition of Non-fact Knowledge

A unit of knowledge that isn't a fact is a kind of **generator**: given fact(s), it may produce other fact(s), as in the figure.

We have already seen knowledge in the form of first-order logic.



Loading [Contrib]/a11y/accessibility-menu.js

Shallow vs. Deep Knowledge

We generally think of knowledge as having **depth**, as in the example. There has always been an attempt to drive rule-based system knowledge deeper, though most rule-based systems I have encountered do not reason as deeply as they could in theory.

Deeper = increased understanding

For example, “dropped objects fall to the ground”

- deeper: “any two bodies in space attract each other”
- deeper: “any pair of entities exist in a space-time curve”

Bayes’ Rule

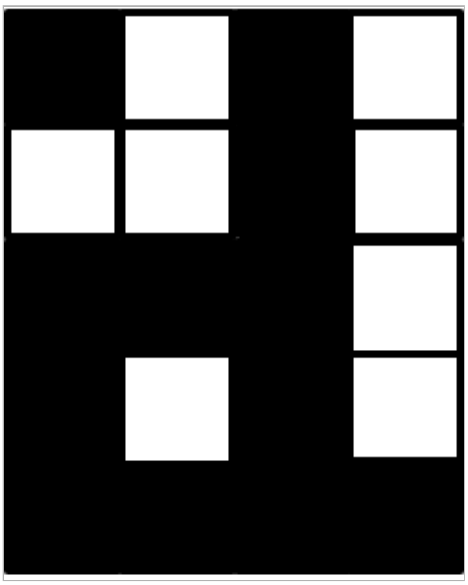
Bayes’ Rule: Example Problem

The example in the figure illustrates the uncertainty involved in deciding whether the element is an “a” or a “b.”

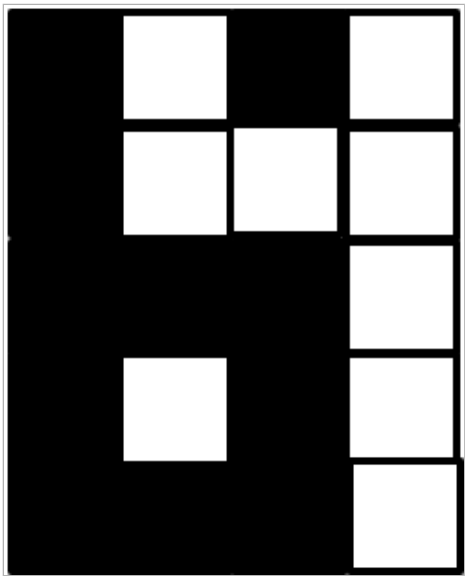
Loading [Contrib]/a11y/accessibility-menu.js

Figure: Deciding Whether the Element Is an *a* or a *b*

Observation 1



Observation 2



Recognize *a* or *b*

Source: Adapted from *Machine Learning* (2nd ed.), by Marsland, p. 27.

Conditional probability concerns the probability of an event when a pertinent fact is known. For example, the probability of selecting a red ball from a bag containing 5 red and 5 blue balls is 50%, but the probability of the same event `given that a blue ball has been removed` is different.

One can rephrase the question

`ervation 1?`

in the figure conditionally as

Loading [Contrib]/a11y/accessibility-menu.js

in the figure conditionally as

```
what is the probability that the letter is 'a', given that 1 has been observed?
```

> Probability that pattern represents 'a' …
> … given observation X …
> Expressed: \(p(\text{letter is }\, a\, | \text{observation}_X)\)

Bayes's Rule: The Goal

The goal: *Find* \(p(a|X)\) *in terms of measurable quantities*. So, probability can be applied to the uncertainty of *if-then* rules by turning them from…

```
IF X THEN a
```

…into…

p(*a*|X), *p* meaning "probability."

In other words, "If I know *X*, how likely is *a* to be valid?"

So far, this has simply been a question of rephrasing. However, now that we are dealing with probabilities, we can invoke a deep, rich theory that has taken over 400 years to develop: *Bayes's Law*.

Bayes's law expresses \(p(A|B)\) in terms of \(p(B|A)\) in case the latter is easier to compute. For example, suppose that we want to know the degree of certainty of the following:

```
IF a person is not motivated to exercise, THEN they are a smoker.
```

To compute this directly, we would have to contact a large group of people who are not motivated to exercise, and count how many of them are smokers. It might be difficult to gather this group in the first place, and when we did, we might find that not every smoker admited to it.

On the other hand, it might be easier to obtain a list of confirmed smokers and ask each of them about motivation.

Bayes's Law allows us to make progress in this way.

> **Use Bayes…**
> …when \(p(A|B)\) is needed but hard to measure, but \(p(B|A)\) is much easier to measure.
>
> Example:
>
> > It may be easier to gather smokers and assess their motivation for exercise \(p(M|S)\) than to gather people with motivational problems and count smokers \(p(S|M)\).

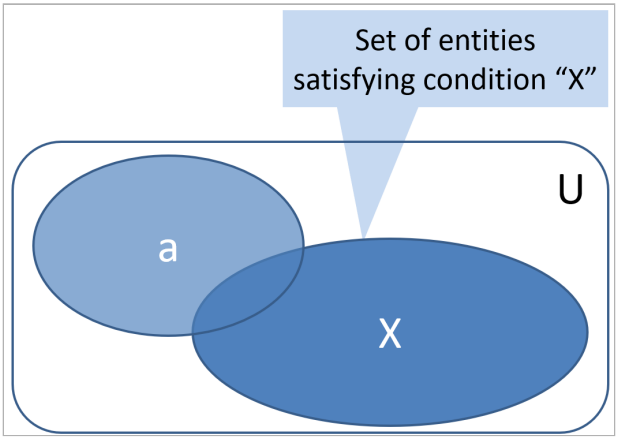
Loading [Contrib]/a11y/accessibility-menu.js

Bayes's Rule: Intuitive Derivation

To understand why Bayes' Law is true, imagine the occurrences satisfying condition \(a\) and those satisfying \(X\), as in the figure. \(U\) here denotes the set of all relevant entities (the "universal" set).

Figure: Entities Satisfying Condition \(a\) and Some Satisfying \(X\)

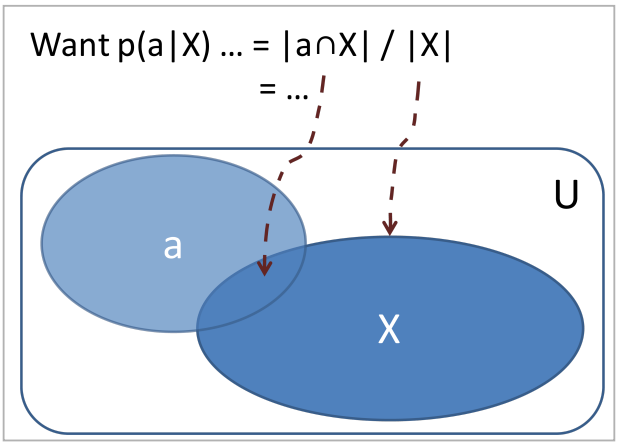
Want \(p(a|X)\)



To simplify, we'll assume that \(a\) and \(X\) are finite, so we can find their size (denoted |…|). If we are given \(X\), the probability of \(a\) is computed by counting entities in \(X\), as shown.

In effect, when you consider \(p(a|X)\), \(X\) is effectively the new universal set.

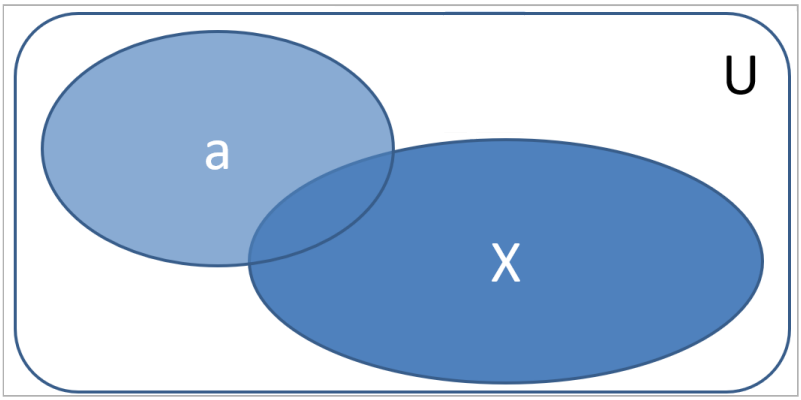
Figure: Entities Satisfying Condition \(a\) and Some Satisfying \(X\) – \(a\) and \(X\) are Finite



By multiplying this quotient by \(|a|\) and dividing by the same amount (thus changing the form but leaving the value unchanged), we obtain Bayes's formula, as in the figure below.

Loading [Contrib]/a11y/accessibility-menu.js

Want \begin{align} p(a|X)\cdots &= \qquad\qquad\;\,|a\cap{X}|\qquad/|X|\\ &=(|a|\qquad\ast|a\cap{X}|/|a|)/|X|\\ &=(|a|/|U|\,\ast|a\cap{X}|/|a|)/|X|/|U|\\ &=\quad{p(a)}\;\;\,\ast p(X|a) \quad\quad/\quad{p(X)} \end{align}



Bayes's Rule

p(C|X) = $p(C)p(X|C)/p(X)$ ⟵ "prior" probabilities

The expressions on the right are known as prior probabilities because they can be computed separately, "before" we compute the conditional probability that we seek.

Here is another example: It is 11 PM and I observe that it is raining. I want the probability that it will rain during my 7 AM commute tomorrow. Suppose that this probability has not been sought in the past, so we don't know it. But suppose that work has been done on the opposite probability: the fraction of times, when it rains at 7 AM, that it was also raining at 11 PM the previous night.

Overall, assume that the following probabilities have already been computed:

- That it rained the previous night at 11 PM, given that it rains at 7 AM
- That it rains at 11 PM (i.e., in general)
- That it rains at 7 AM

p(C|X)=p(C)p(X|C)/p(X)

↑

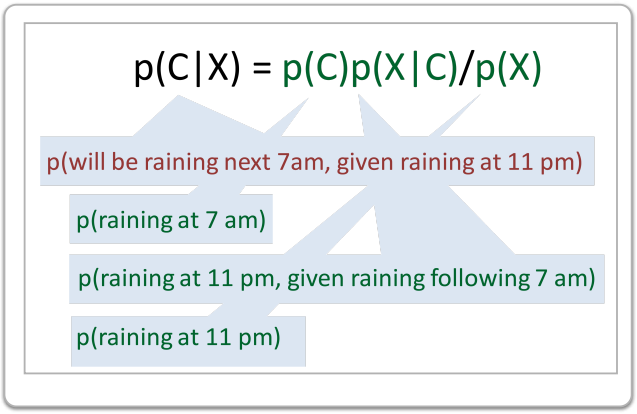
p(will be raining next 7 AM, given raining (now) at 11 PM)

Assume that we don't have the data for this, but we do have…

p(raining at 11 PM, given raining following 7 AM)

Loading [Contrib]/a11y/accessibility-menu.js

Knowing these (in green—*p(C)p(X|C)/p(X)*), we can compute the desired probability (in red—p(will be raining next 7 AM, given raining (now) at 11 PM)).



Application Examples

Blockchain

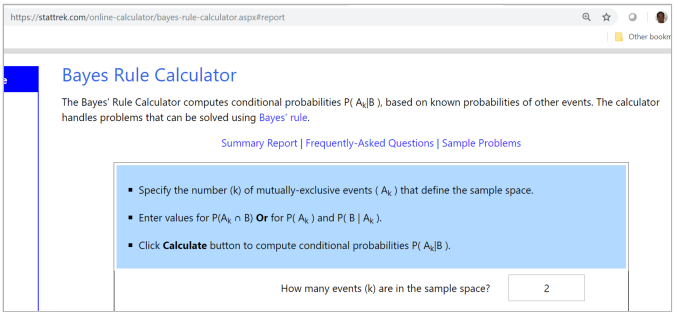
Estimate *demand* (D) for a product using "block count" (*mention* (M) of its SKU in a blockchain). The key prior here is how frequently the product is mentioned under various demand conditions. Demand can be measured by sales figures.

$$p(D|M) = p(D)\, p(M|D)/p(M)$$

Online Bayes Calculators

The Bayes calculator in the figure allows any number (k) of conditional events, and allows the entry of all $P(B|A_k)$'s —or of $P(A_k \cap B)$, from which $P(B|A_k)$ can be calculated as in the derivation shown previously of Bayes' rule.

Figure: Bayes Rule Calculator



Source: https://stattrek.com/online-calculator/bayes-rule-calculator.aspx

The figure shows the sets involved in the calculation. In this example 1, forty percent of B elements are A elements.

Figure: Bayesian Calculator Example 1

Loading [Contrib]/a11y/accessibility-menu.js

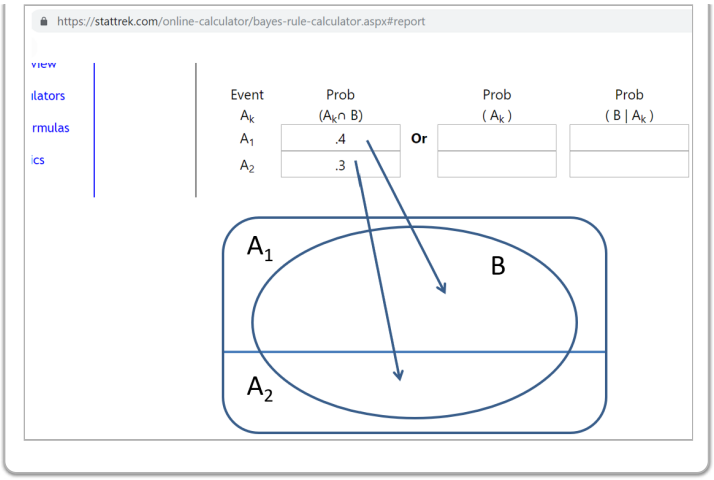
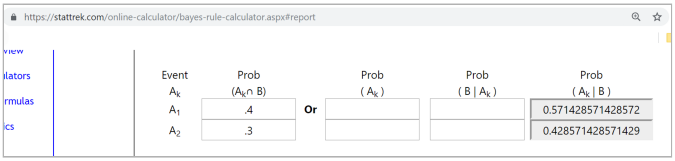


Figure: Bayesian Calculator Example 1 Execution



Source: https://stattrek.com/online-calculator/bayes-rule-calculator.aspx

In the following example 2, we have the actual prior conditional probabilities.

Figure: Bayesian Calculator Example 2

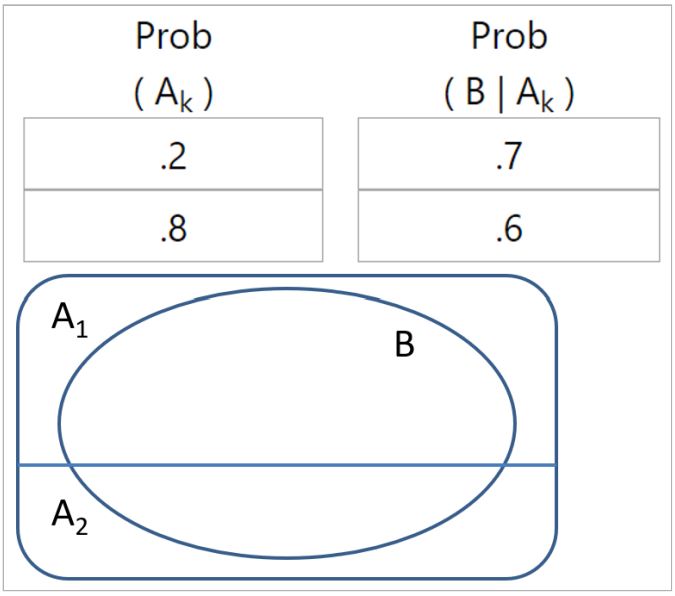
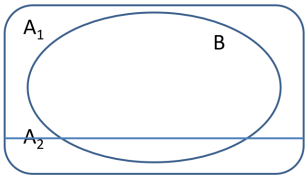


Figure: Bayesian Calculator Example 2 Execution

Loading [Contrib]/a11y/accessibility-menu.js

Event A_k	Prob $(A_k \cap B)$		Prob (A_k)	Prob $(B \mid A_k)$	Prob $(A_k \mid B)$
A_1		Or	.2	.7	0.225806451612903
A_2			.8	.6	0.774193548387097



Source: https://stattrek.com/online-calculator/bayes-rule-calculator.aspx

Product Rule

$p(A\underbrace{\text{ and }}_{\text{ Also denote } p(A,B)}B)=p(B|A)p(A)$

The expressions on the right are known as prior probabilities because they can be computed separately, “before” we compute the conditional probability that we seek.

Using Independent Variables

Bayes’ law with events X and A essentially measures “X and A).” The common use of Bayesian reasoning is to compare outcomes based on multidimensional data by using the more general form of Bayes’ rule shown in the figure (in this case, for 3 dimensions).

Suppose that A, B, and C are observable events such as A = “customer browsed chairs in the past 2 months,” B = “customer bought a house within the past year,” and C = “customer browsed fabrics in the past 2 months.” We want to know how likely it is that the customer is interested in a couch, or an arm chair etc. We base this on the fraction (probability) of those customers who actually bought a couch had previously browsed chairs etc.

The Bayesian quantity $p(X)\times{p(A|X)}\times{p(B|X)}\times{p(C|X)}$ measures X as a probable outcome, in the presence of three other events.

Bayes:

$p(X)\cdot p(A|X) = p(A)\cdot p(X|A)$
measures X and A.

More generally, measure:

$p(X)\times{p(A|X)}\times{p(B|X)}\times{p(C|X)}$

- $p(X)$: with no classification knowledge
- $p(A|X)$: with more classification knowledge

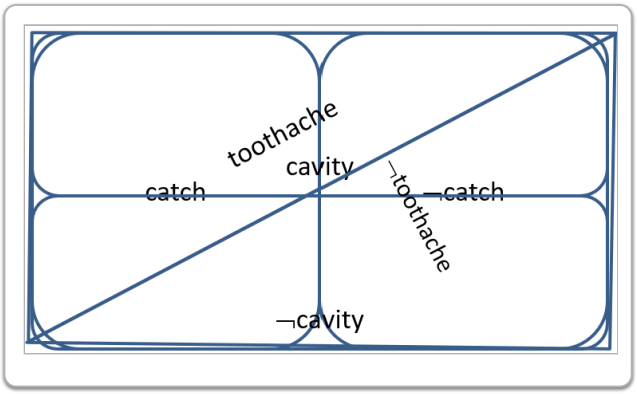
“prob of X and A-given-X and B-given-X)…”

Loading [Contrib]/a11y/accessibility-menu.js

Cavity Example: Given (a priori) Probabilities

Using Bayes in reality is an arithmetic-intensive process in which the probability of every possible outcome is computed. The table pertains to data about a patient, whose probability of having a cavity is computed. This is based on eight priors such as `the probability of (anyone) with a cavity having a toothache, as well as a "catch" is 0.108`.

A Full Joint Distribution for the *Toothache*, *Cavity*, *Catch* World				
	Toothache		**\(\lnot\) *Toothache***	
	catch	**\(\lnot\) *catch***	***catch***	**\(\lnot\) *catch***
cavity	0.108	0.012	0.072	0.008
\(\lnot\) cavity	0.016	0.064	0.144	0.576



The summation formula indicates adding all of these joint probabilities because the events are disjoint, and comprise all possibilities. We will apply this pattern several times in what follows.

$$P(Y)=\displaystyle{\sum_{z\in{Z}}}P(Y,z)$$

$$P(\text{cavity})= 0 .108 + 0.012 + 0.072 + 0.008 = 0.2 $$

Another decomposition form: conditioning. Bayes formula for multiple events also computes a probability, as shown in the following formula:

$$P(Y)=\displaystyle{\sum_{z}}P(Y|z)P(z)$$

Bayes allows us to answer a wide variety of combination questions such as `the probability that a person with a toothache has a cavity`. The figure shows a form of Bayes with an AND instead of the equivalent conditional. The sums follow from the logical completeness of the predicates (e.g., *catch* AND \(\lnot\)*catch*).

Loading [Contrib]/a11y/accessibility-menu.js

A Full Joint Distribution for the *Toothache*, *Cavity*, *Catch* World				
	Toothache		**\(\lnot\) *Toothache***	
	catch	**\(\lnot\) *catch***	***catch***	**\(\lnot\) *catch***
cavity	0.108	0.012	0.072	0.008
\(\lnot\) *cavity*	0.016	0.064	0.144	0.576

\(\begin{align} P(cavity\,|\,toothache)&=\frac{P(cavity\land{toothache})}{P(toothache)}\\ &=\frac{0.108+0.012}{0.108+0.012+0.016+0.064}=0.6 \end{align}\)

Just to check, we can also compute the probability that there is no cavity, given a toothache:

\(\begin{align} P(\lnot{cavity}\,|\,toothache)&=\frac{P(\lnot{cavity}\land{toothache})}{P(toothache)}\\ &=\frac{0.016+0.064}{0.108+0.012+0.016+0.064}=0.4 \end{align}\)

Examples

In this section we present an example of Bayesian “reasoning.”

Example (Geology)

The example is to predict which of *dolomite* or *shale* is more likely given a gamma ray reading.

- Quest: dolomite or shale?
- Observed: Gamma ray reading >60 or not

The following shows the symbolic formulation of the problem.

Using Bayes' rule, we can determine the posterior probability of occurrence of dolomite and shale, given that we have actually observed a gamma ray value greater than 60. Let's define event & probabilities as follows:

\(A\): *GammaRay* > 60

\(B_1\): occurance of dolomite

e of shale

Loading [Contrib]/a11y/accessibility-menu.js

Seek $P(B_1|A)$

Source: https://www.yumpu.com/en/document/read/19083769/applications-of-bayes-theorem

All of the priors are shown here: in other words, the probabilities that can be computed. For example, $P(A|B2)$ is obtained by subjecting (known) shale to gamma rays.

$P(B_1)$: prior probability for dolomite based on overall prevalence 60% (476 of 771 core samples)
$P(B_2)$: prior probability for shale based on overall prevalence 40% (295 of 771 core samples)
$P(A|B_1)$: prior probability of *GammaRay* > 60 in a dolomite=7% (34 of 467 dolomite samples)
$P(A|B_2)$: prior probability of *GammaRay* > 60 in a shale=95% (280 of 295 shale samples)

Source: https://www.yumpu.com/en/document/read/19083769/applications-of-bayes-theorem

We will be comparing the probability of shale vs. dolomite, and the probability in the figure is a common denominator, so we can normalize as discussed, and so computing this is not really necessary.

The the denominator in Bayes' theorem, the total probability of A, is given by

$$\begin{align} P(A)&=P(A|B_1)P(B_1)+P(A|B_2)P(B_2)\\ &=0.07\times{0.60}+0.95\times{0.40}=0.422 \end{align}$$

Source: https://www.yumpu.com/en/document/read/19083769/applications-of-bayes-theorem

Now we apply Bayes to compute the probability of each of the two rock types give that a gamma ray reading greater than 60 was observed.

If we measure a gamma ray value greater than 60 at a certain depth in a well, then the probability that we are logging a dolomite interval is

$$\begin{align} P(B_1|A)&=\frac{P(A|B_1)P(B_1)}{P(A)}\\ &=\frac{0.07\times{0.60}}{0.422}=0.10 \end{align}$$

and the probability that we are logging a shale interval is

$$\begin{align} P(B_2|A)&=\frac{P(A|B_2)P(B_2)}{P(A)}\\ &=\frac{0.95\times{0.40}}{0.422}=0.90 \end{align}$$

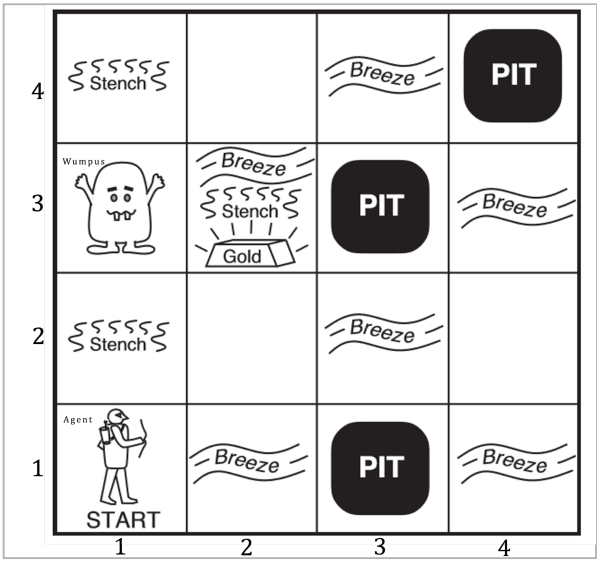
Loading [Contrib]/a11y/accessibility-menu.js

Thus, our observation of a high gamma ray value has changed our assessment of the probabilities of occurrence of dolomite and shale from 60% and 40%, based on our prior estimate of overall prevalence, to 10% and 90%.

Source: https://www.yumpu.com/en/document/read/19083769/applications-of-bayes-theorem

Example: Wumpus World

We now move a bit closer to the complexities of the real world (even though we'll stay with Wumpus World).



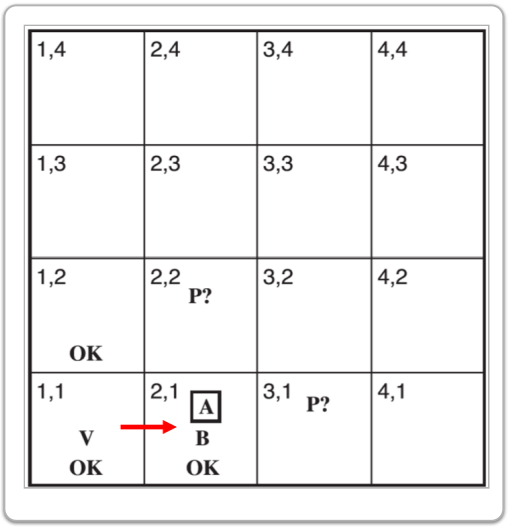
In squares adjacent to:

a pit: a breeze

a wumpus: a stench

Logical Reasoning in Wumpus World

Loading [Contrib]/a11y/accessibility-menu.js



We have seen that, to guide the hero's actions, logic is a sound approach. But this is in theory—and pure logic requires all of the facts. We rarely have all of the facts when faced with having to act. For example, if the hero is in (position) 1.1, they have all the necessary knowledge to proceed (B=Breeze and P=Pit). But after that, my appropriate moves from 2.1 are uncertain because they don't know whether the "breeze" emanates from 2.2 or 3.1.

Suppose agent moves (⟶) [1,1] to [2,1].

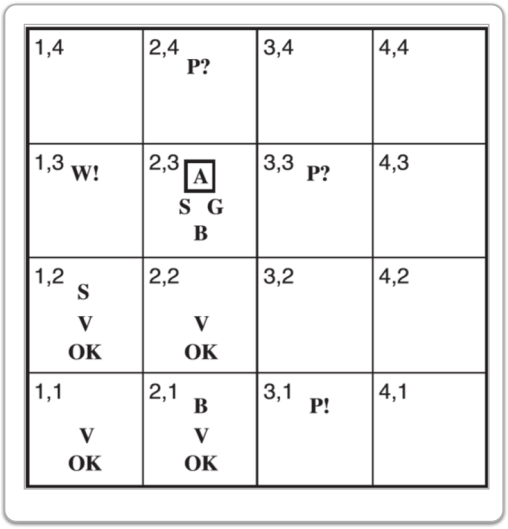
Perceives breeze in [2,1].

Then, delineated the prudent backtracking action.

Must be a pit in neighboring square—[2,2], [3,1] or both.

Prudent agent will go back to [1,1], then to [1,2].

This figure shows the result of purely logical reasoning (P=Pit, S=Smell, W=Wumpus, V=Visited).

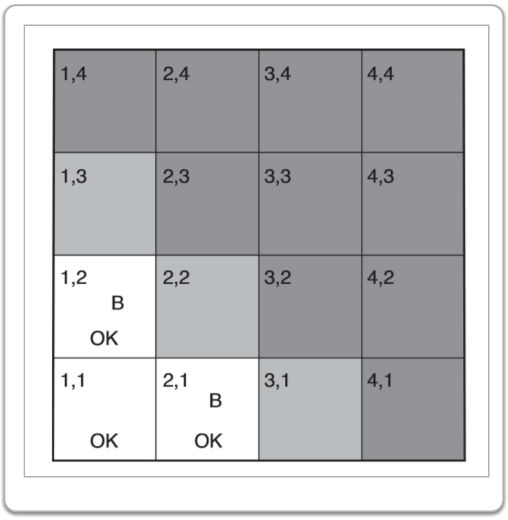


Given the (negative) info:

- Wumpus can't be in [1,1]
- can't be in [2,2]
- thus (…) wumpus in [1,3]
- No breeze in [1,2] implies no pit in [2,2].
- Thus pit in [3,1].

But Agent's sensors only partial global information. The figure describes the current uncertainty about the next move to the three gray squares. Each reachable square—[1,3], [2,2], and [3,1]—might contain a pit.

Loading [Contrib]/a11y/accessibility-menu.js



We must thus accept and embrace uncertainty, i.e., use probabilistic reasoning (pure randomness being unnecessary).

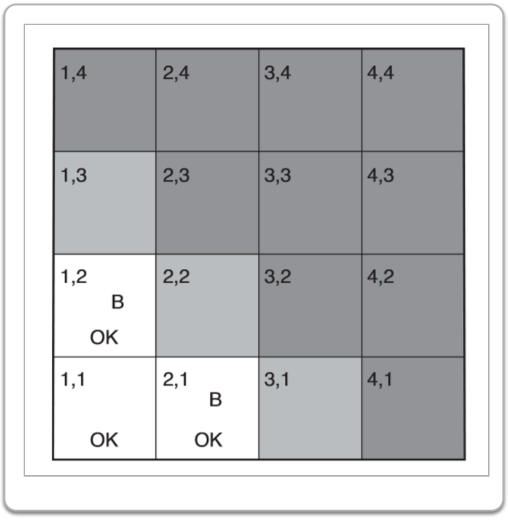
Logical vs. Probabilistic

Pure logical inference based on incomplete information can conclude nothing about which square is most likely to be safe.

… might have to choose randomly.

Probabilistic agent is much better.

The following shows the conversion of the uncertain situation into a probabilistic statement, specifically including \(P(\text{Position 1.3 contains a pit} | (\text{known}\land{b})\).



In this situation →

know …

- observed breeze (T or F) in visited (white) squares +
- each contains no pit.

Abbreviate these facts as \(b=\lnot{b_{1,1}}\land{b_{1,2}}\land{b_{2,1}}\) and \(known=\lnot{p_{1,1}}\land\lnot{p_{1,2}}\land\lnot{p_{2,1}}\)

We are interested in answering queries such as \(P(P_{1,3}|known,b)\): how likely is it that [1,3] contains a pit, given the observations so far?

Here let's apply joint probability approach again. The figure below recalls the approach used to calculate \(P(\text{Cavity } | \text{ toothache})\), which converts it to \(P(\text{Cavity AND toothache})\) with normalization factor \(\alpha\) (which we will not need to calculate because we are doing comparisons with the results for each course of action).

Loading [Contrib]/a11y/accessibility-menu.js

Let unknown be the set of $P_{i,j}$ variables for squares other than the known squares and the query square[1,3].

Recall:

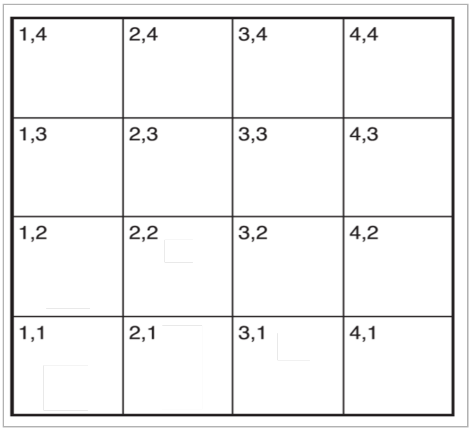
$\begin{align} &P(Cavity\,|\,toothache)\\ &=\alpha{P(Cavity,toothache)}\\ &=\alpha[P(Cavity,toothache,\bbox[5px,border:2px solid red]{catch})+P(Cavity,toothache,\bbox[5px,border:2px solid red]{\lnot\,{catch}})] \end{align}$

Note: $\bbox[5px,border:2px solid red]{catch}$ and $\bbox[5px,border:2px solid red]{\lnot\,{catch}}$: Range over all truth values of all subsets of any particular set of RV's.

Similarly,

$\mathbf{P}(P_{1,3}\,|\,known, b)=\alpha\displaystyle{\sum_{unknown}}\mathbf{P}(P_{1,3},unknown,known,b)$

More generally, identify random variables and use joint probabilities. We actually need, for every square, the probability that it contains a pit as well as whether it is breezy. That would enable us to make trade-offs in moving.



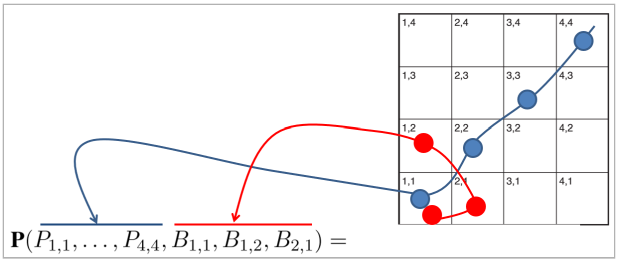
Define Boolean variables P_{ij} and B_{ij} for each square:

$P_{ij} \text{ TRUE } \equiv {\text{square }[i,j]} \text{ contains a pit}$

$B_{ij} \text{ TRUE } \equiv {\text{square }[i,j]} \text{ breezy}$

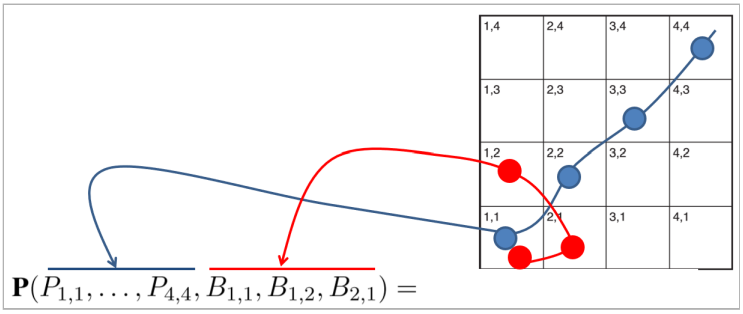
Identify desired probabilities in the Wumpus World example. The figure expresses the probability distribution for pits at all locations (only the diagonal ones are shown) together with breezes at the three places of interest.

Probability of pits are as shown in red, and probability of breezes are as shown in blue.



Loading [Contrib]/a11y/accessibility-menu.js

Identify random variables and manipulate joint probabilities: The desired joint probability distribution can be decomposed into the product shown.



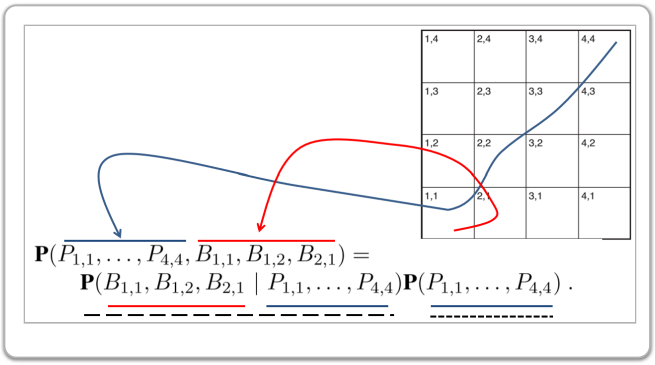
\(\mathbf{P}(P_{1,1}\ldots,P_{4,4},B_{1,1},B_{1,2},B_{2,1})=\)
\(\mathbf{P}
(B_{1,1},B_{1,2},B_{2,1}|P_{1,1}\ldots,P_{4,4})\mathbf{P}
(P_{1,1}\ldots,P_{4,4})\) ← Product rule

Boolean variables \(P_{ij}\) and \(B_{ij}\) for each square:

\(P_{ij} \text{ TRUE } \equiv {\text{square }[i,j]} \text{ contains a pit}\)

\(B_{ij} \text{ TRUE } \equiv {\text{square }[i,j]} \text{ breezy}\)

Loading [Contrib]/a11y/accessibility-menu.js



First term is the conditional probability distribution of a breeze configuration, given a pit configuration; its value are 1 if the breezes are adjacent to the pits and 0 otherwise.

The second term is the prior probability of a pit configuration. Each square contains a pit with probability 0.2, independently of the other squares; hence,

\(\mathbf{P}
(P_{1,1}\ldots,P_{4,4})=\displaystyle{\prod^{4,4}}\mathbf{P}
(P_{i,j})\).

For a particular configuration with exactly \(n\) pits, \
(P(P_{1,1}\ldots,P_{4,4})=0.2^n\times{}0.8^{16-n}\).

The second term of the product expresses a probability distribution (function) but we can compute it for each occurrence of pits. The calculation is for \(n\) of the locations to be pits and the rest not pits. The occurrences are independent (a pit at one location has no influence on whether there is a pit at another).

Summary: Uncertainty

- Use expert's rules if possible
- Use Bayes when priors can be computed
  - Mathematical/statistical
  - Computationally intensive
  - Heuristics available

Fuzzy Logic

Loading [Contrib]/a11y/accessibility-menu.js

Learning Objectives

What can we expect machines to learn? Much of what needs to be learned carries uncertainty. For example, \(X\) is a dining room because it contains a table and several chairs.

- Fuzzy is a useful approach to uncertainty.
- Key motivation for fuzzy: humans' use of language.

After successfully completing this part of the module, you will be able to do the following:

1. Exploit fuzziness in data.
2. Create fuzzy rules.
3. Apply fuzzy rules.

Introduction and Fuzzy Linguistic Variables

Advantages over Conventional Expert System (ES)

As we will see, fuzzy expert systems tend to need fewer rules than conventional ES's. Fuzzy rules can operate in parallel, and their "fuzziness" can make them less brittle than conventional rules.

- Fewer rules
- Easier to express
- Amenable to parallelism
- More robust

Fuzzy Disadvantages

The main disadvantage of fuzzy rules is that they do not facilitate chains of reasoning. The other listed disadvantages concern the fact that they are empirical. However, this is true for almost all machine learning techniques—with the possible exception on Bayesian reasoning. Also, automated chain reasoning from data seems to be a long way off in any case.

- No chaining
- No guarantee of stability
- Few theoretical results
- Verification requires extensive testing

Fuzzy disadvantages summary is adapted from Munakata & Jani, CACM, March 1995

When to Apply Conventional/Fuzzy

Loading [Contrib]/a11y/accessibility-menu.js

The following bullet points contrast examples of conventional vs. fuzzy expert systems. The operator of a semitrailer uses a relatively small set of rules ("If I am in … position and … then I take … action") rather than a rule that concludes an intermediate statement, and thus facilitates chains of reasoning in which one application of a rule produces results that are then used in another rule application.

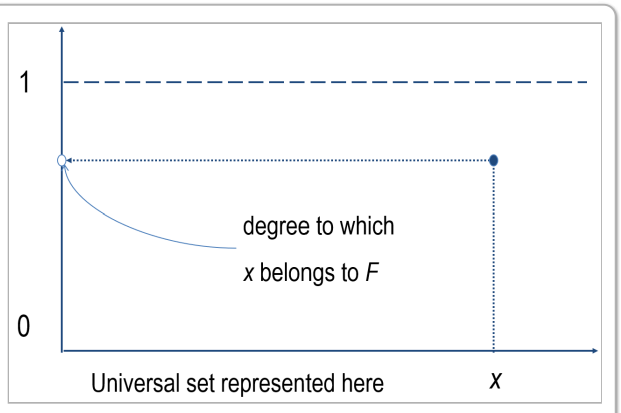
- Logically deep: Conventional
  - e.g. radar repair
  - e.g. business prognosis with multiple branching
- Small number of rules: Fuzzy
  - e. g., semi-trailer backup
  - e.g. rough business prognosis

Common Visualization of Fuzzy Set \(S\)

A set in mathematics is something for which it is possible to state, for anything, whether that thing belongs or not. An example is the set of chairs in Manhattan on January 1, 2017 at 3 pm. The universal set is the set consisting of everything.

Lotfi Zadeh, a prominent electrical engineer, speculated in the 1950's about the mismatch between the concreteness of sets and how this mismatched the real world. He invented the idea of a fuzzy set (let's call an example \(F\) for which one can state, for anything, *the degree to which* that thing belongs to \(F\).

The figure shows a way to visualize a fuzzy set in two dimensions. It imagines the universal set as spread out on the \(x\) axis, and a typical element \(x\). The degree to which \(x\) belongs to \(F\) is represented by a point whose \(y\) coordinate is between 0 (equivalent to not belonging at all in the conventional sense and 1 (belonging entirely). Since the sets we will be concerned with involve only numbers, this visualization proves to be adequate.

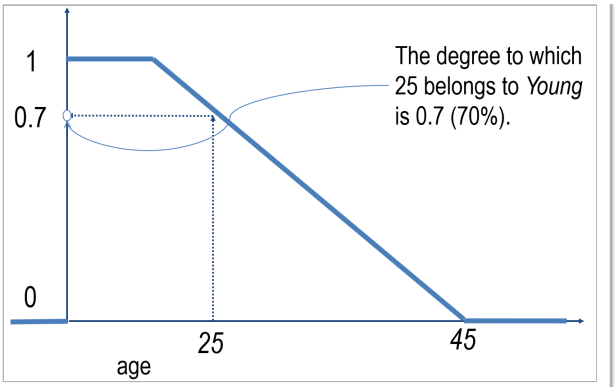


Source: http://cstms.berkeley.edu/current-events/on-a-history-of-fuzzy-sets-and-systems/

The definition of a ***fuzzy set*** depends on the context. For example, in the context of a nursery school application, a 10-year-old would belong the the fuzzy set *Young* to the degree 0. For an insurance application, the fuzzy set *Young* might be defined as in the figure. In this particular definition, the number -1 is not considered to belong to this set—more properly, we say that its degree of belonging to *Young* is zero.

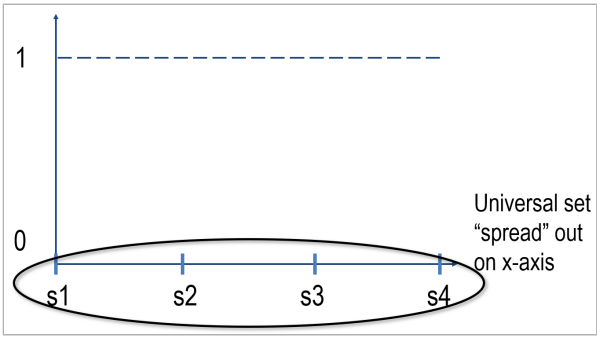
Loading [Contrib]/a11y/accessibility-menu.js

Example: The Fuzzy Set *Young*



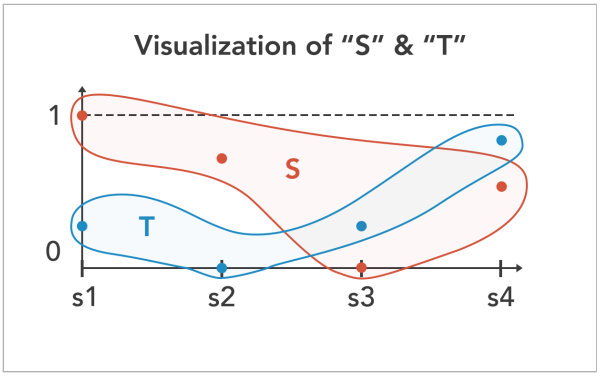
As an exercise in reinforcing the fuzzy idea, the figure concerns a system in which the universal set consists of just four elements, labeled s1, s2, s3, and s4. We imagine them as four positive numbers.

Visualization of Fuzzy Sets



The following figure shows two fuzzy sets in this space. There is an extensive theory about intersecting fuzzy sets etc. but we will not need most of it for machine learning in this course.

Visualization of "S" & "T"



Fuzzy Linguistic Variable

In fuzzy work, we often work with variables (such as `Temperature`) whose values are fuzzy sets (such as `VeryLow`, `Low`, Java of type `String`—the values of s are strings. Variables whose values are fuzzy sets

Loading [Contrib]/a11y/accessibility-menu.js

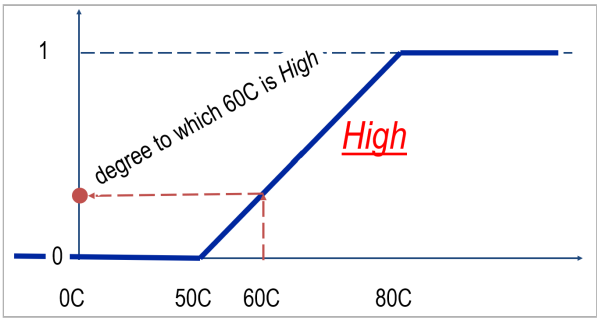
are called ***fuzzy linguistic variables***.

> A variable whose values are fuzzy sets over a common range of numbers. Example: `Temperature`, with values `VLo, Lo, Med, Hi, VHi`.

Fuzzy Linguistic Variable Temperature

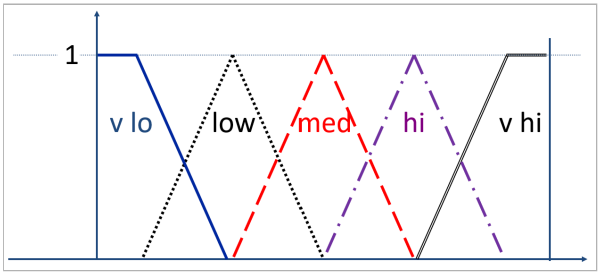
One of its values: High

Here is a picture of the fuzzy set High, a value of the fuzzy linguistic variable `Temperature`.



Common Fuzzy Values

The figure shows the general shape of commonly used fuzzy values.



Common Place for Fuzzy Use

The layman often imagines that fuzzy expert systems are used when the input or output is fuzzy but this is not generally true: it is the **model** that expresses fuzziness. Inputs and outputs are more often crisp.

Input → **Model** → **Output**

- Input: e.g., pendulum position/velocity
- Model: Decision mechanism
- Output: Which way to move

Fuzzy Rules

Now that we have defined fuzzy linguistic variables, we can define rules with them in the same way that we defined expert system rules. Recall that the latter are like

```
IF strength is >80 units and speed is >70 units THEN recruit is promising.
```

For classical expert systems, this refers to values of strength and speed—ordinary, (“crisp”) variables. This example carries

Loading [Contrib]/a11y/accessibility-menu.js

`trength` and `Speed` whose values are fuzzy sets like `Fast`, `Slow`, `High Low`, etc., as

explained in this section.

Fuzzy Control Model

The form of a fuzzy rule is showing below. Notice that we are not allowing for an `OR` conclusion.

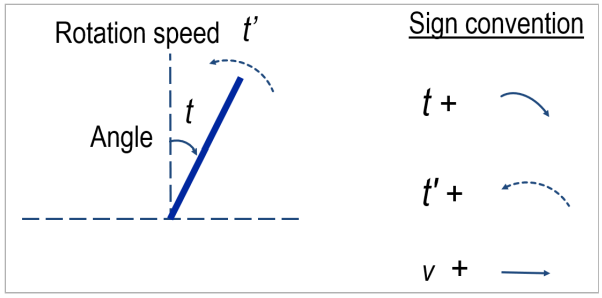
Made up of controller rules (model)

```
IF    <fuzzy variable> IS <fuzzy value>

AND   <fuzzy variable> IS <fuzzy value>

…

THEN  <fuzzy variable> IS <fuzzy value>
```

The classical example of a fuzzy rules set is one that balances a stick (formally referred to as an “inverted pendulum”). Three fuzzy variables are involved:

- `Angle` (i.e., with the vertical), shortened to t,
- `Angular Velocity` (e.g., measured in degrees per second), shortened to t’, and
- `Velocity` (e.g., measured in degrees per second), shortened to t’.

The figure shows the directions that are taken as positive and negative.



Look at an example of a fuzzy rule. You can check for yourself that if the pendulum (stick) were leaning to the right and rotating clockwise fast, you would need to move the bottom of the stick to the right at high speed in order to keep it balanced.

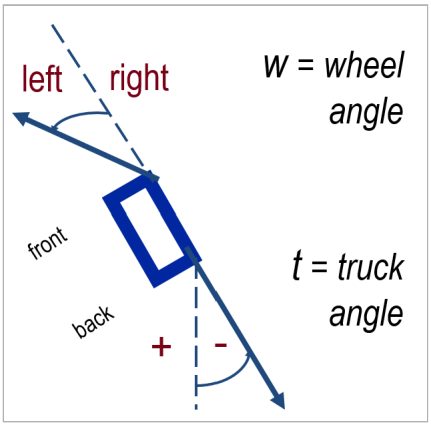
Example:

```
IF    t is Positive Large AND t' is Negative Large

THEN  v Positive Large
```

Loading [Contrib]/a11y/accessibility-menu.js

The idea of fuzzy linguistic rules codifies useful rules-of-thumb such as those used when backing up a truck to a loading bay—or even more complex ones such as backing up a semi-trailer.

Truck Backing Example




```
IF
      location left quadrant
&     t  negative large
&     w left medium

THEN
      turn wheel clock large
```

The fuzzy values of one fuzzy linguistic variable are not necessarily the same as those of another; however, they often have names such as "`Large`." The "`Large`" fuzzy value for one variable is unlikely to be the same as for another, though.

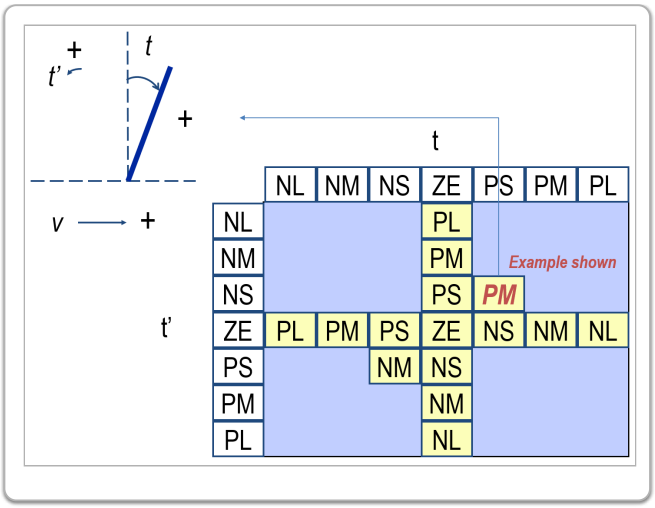
In the inverted pendulum example, \(t\) and \(t'\) share values with the same names, as in the figure. Positive Small, for example, has different for \(t\) as for \(t'\), however.

Pendulum Values for \(t\) and \(t'\)

```
NL      Negative Large
NM      Negative Medium
NS      Negative Small
ZE      Zero
PS      Positive Small
PM      Positive Medium
PL      Positive Large
```

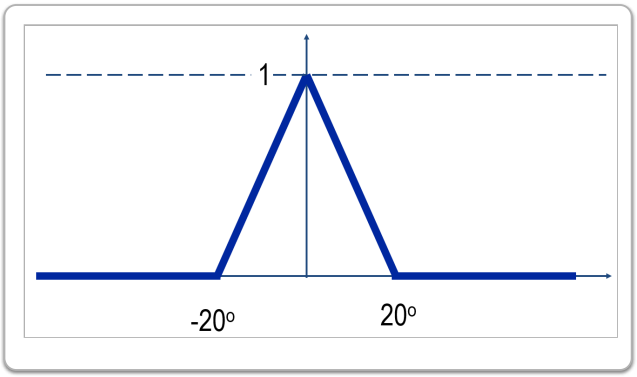
The rules of a fuzzy expert system with two variables can be conveniently expressed by a matrix, as in the figure. For example, the rule `PM` is "`if \(t\) is Positive Small and \(t'\) is Negative Small (i.e., counter-clockwise), then velocity is Positive Medium.`" This particular rule is illustrated in the figure.

Loading [Contrib]/a11y/accessibility-menu.js



Fuzzy Value Example: Zero (ZE)

Consider the (fuzzy set) value `Zero` of a `Temperature` variable. Certainly, \(0^o\) qualifies fully as belonging to `Zero`. On the other hand, \(50^o\) almost certainly does not. One version of `Zero` is shown in the figure.



Applying Fuzzy Rules to Crisp Input

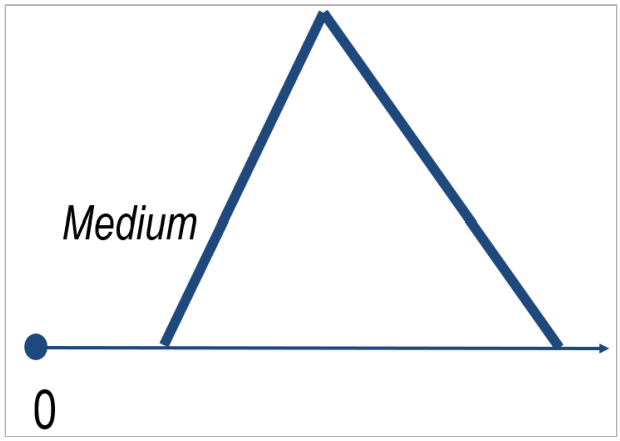
We can express fuzzy rules. In this section, we will describe what a fuzzy expert system does with an input. The inputs will be assumed to be an ordinary numbers. The term ***crisp*** is used to emphasize this.

Applying “A => B” to Input: Example

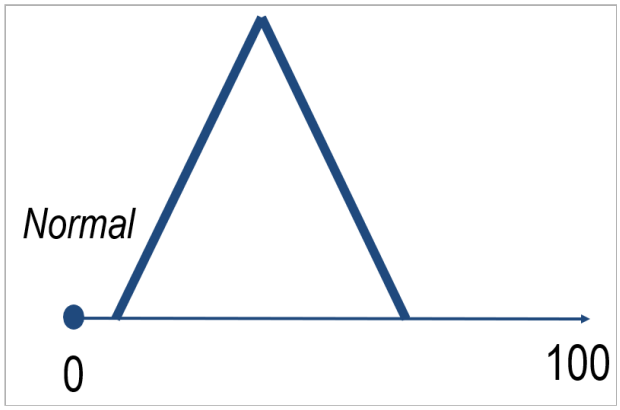
Imagine a rule for which the input (the `IF` part) is `Temperature` and the output (`THEN` part) is `Alert Level` on a scale from 0 to 100.

Loading [Contrib]/a11y/accessibility-menu.js

A: Temp == Medium



B: Alert Level == Normal

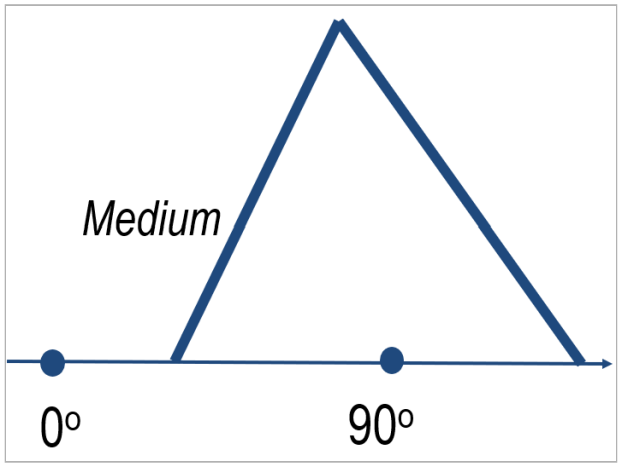


Implementing "\(A=\gt{B}\)" to Input: Example

Suppose that we have the rule "`IF Temp == Medium THEN Alert Level == Normal`." (We are using '==' to emphasize that the variable has the value, although we usually use just '='.)

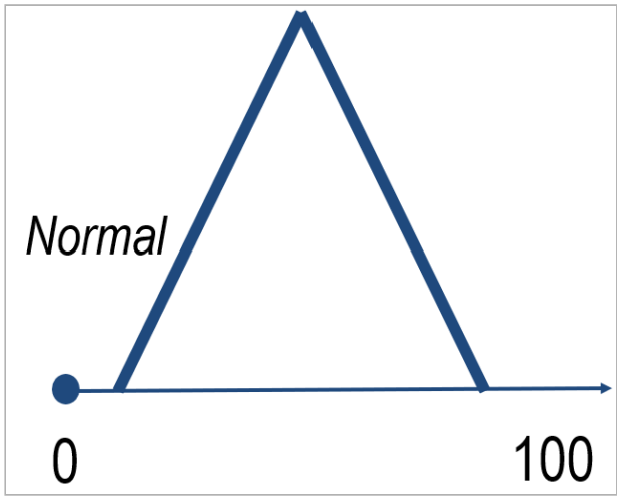
Suppose that, in addition to this rule, we also know that the temperature is measured at \(90^o\). The question is *what does the fuzzy system do with crisp input?*

Temp



Given: furnace temp \(90^o\) …

Alert Level



Consequent value via rule?

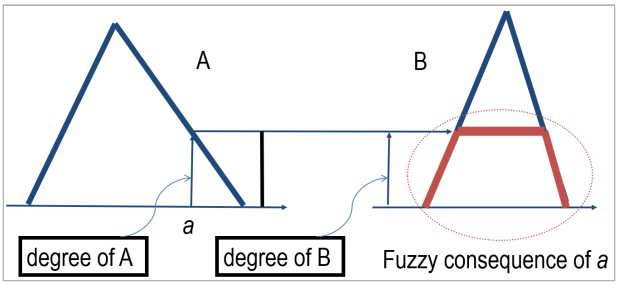
When a crisp value \(a\) is observed, it conditions the premise \(A\) (`IF` part); in other words \(a\) and \(A\) determine the extent to which the premise is true. We are asking *to what extent is \(B\) true?* It is natural to say that `if` \(A\) `implies` \(B\) `and` \(A\) `is true some particular extent then` \(B\) `is true to the same extent`. That is what the

Loading [Contrib]/a11y/accessibility-menu.js

ent to which the fuzzy set \(B\) is true.

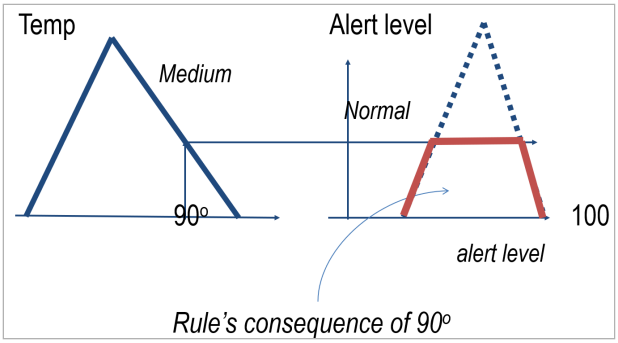
Implementing "\(A=\gt{B}\)"

Given: a crisp value in range of \(A\)



The consequence of the input \(90^o\) is thus the fuzzy value shown in red. It is probably not equivalent to any single name like "large," "somewhat large" etc. but it is nevertheless well-defined.

Implementing "\(A=\gt{B}\)"

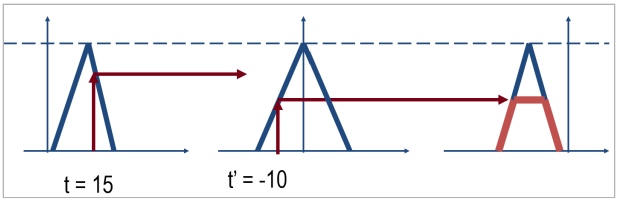


The following figure shows the consequence (in red) of

- The rule `IF t = PS AND t' = ZE THEN v = NS`
- The input `t = 15`, and
- The input `t' = -10`

Implementing "\(A\&{B}=\gt{C}\)"

```
IF t = PS AND t' = ZE THEN v = NS
```

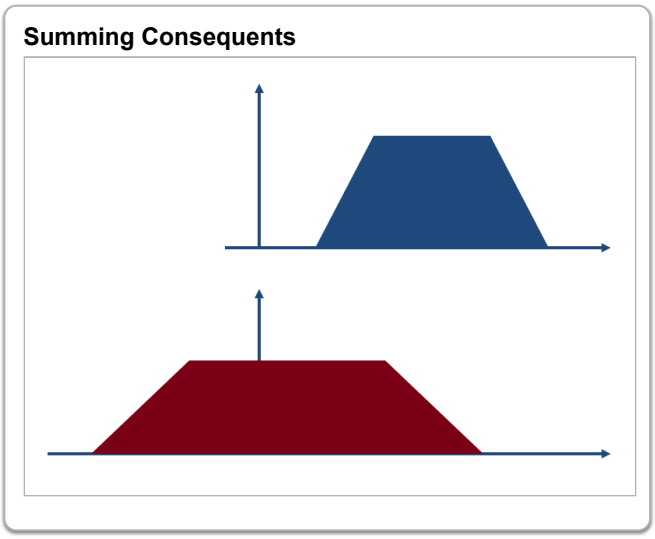


Summing Consequents

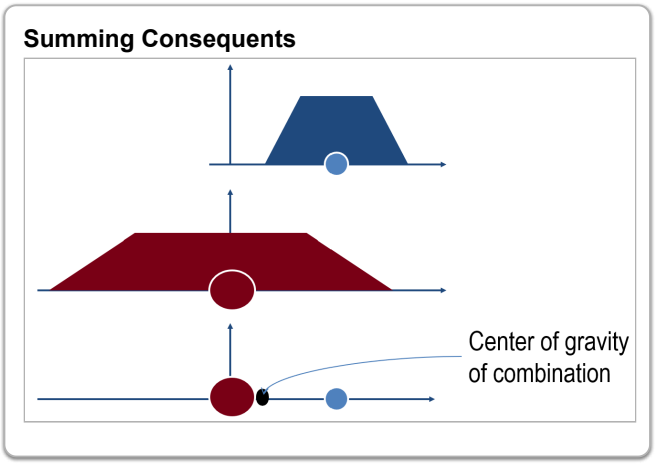
Applying rules to input thus produces a set of fuzzy (set) values—often trapezoidal in shape. What we actually want is a concrete ("crisp") result. This is usually done by treating the fuzzy sets as, in effect, metal plates* and selecting as the crisp

Loading [Contrib]/a11y/accessibility-menu.js

value, the center of gravity (“centroid”) of the result.



To find the center of gravity of a set of shapes, you can replace each with a weighted point at its own center of gravity. Then you find the weighted average of the point weights. This is demonstrated in the figure.

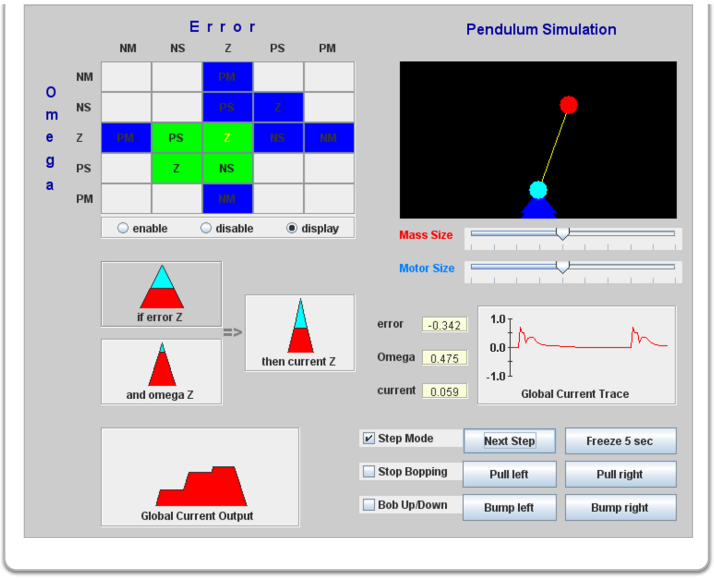


Demonstration

The figure “Fuzzy Pendulum Demonstration” shows the rules (in green) that provide nonzero contribution to the velocity of the base (actually, the torque, which is equivalent here). You can select a single rule—Zero and Zero implies Zero in the figure—that shows the way that rule contributes.

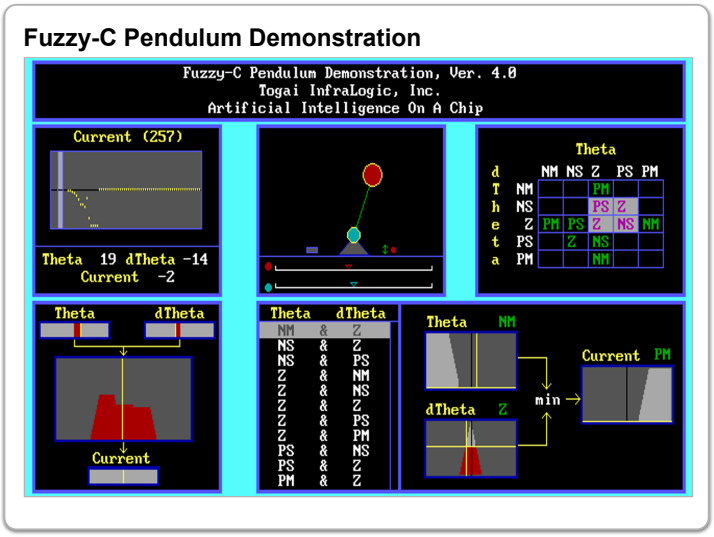
The pendulum is “knocked” now and then to get it going.

Loading [Contrib]/a11y/accessibility-menu.js



Watch video Inverted Pendulum.

The following figure shows essentially the same example. The current status of the rule Theta (we called it \(t\)) NM (negative medium) AND dTheta (we called this \(t'\) Z (zero) IMPLIES Current (equivalent to velocity) PM (positive medium).



Online Demonstrations: There are various demonstrations of fuzzy control online, as in the list below, browsers generally don't support. These are good for visualization, although our main interest for machine learning is to extract fuzzy rule so that they can be used for and to explain predictions base on the data.

- Fuzzy Shower demo – java only code (simple shower control simulation)
- Fuzzy Shower demo – Java/Jess hybrid code (simple shower control simulation)
- Fuzzy Compiler demo – java only code (tabular output of a set of fuzzy rules run over a range of inputs)
- Fuzzy Truck demo – java only code (park a truck using fuzzy rules)

Loading [Contrib]/a11y/accessibility-menu.js

Example: Supplemental Feeding for Range Cows

The following example shows fuzzy rules for a feedstock application. They are understandable.

- Rule 1: If protein is low and energy is low then feed is high
- Rule 2: If protein is low and energy is high then feed is low
- Rule 3: If protein is high and energy is low then feed is high
- Rule 4: If protein is high and energy is high then feed is low

The source of the example: http://cals.arizona.edu/AREC/fuzzy/example.html

Example: Power System Peak Load Forecasting

The example outlined below supports the advantages of gleaning explainable rules from data. There have been many applications since that time.

- Forecasts daily load curve's two minima and two maxima, for each season.
- Tested using historical load and temperature data of Greek interconnected power system.
- Results: Can forecast future loads with accuracy comparable to neural networks.
- Can also incorporate classic rules and expert's opinion.

The source of the example: A Fuzzy Expert System for Peak Load Forecasting. Application to the Greek power system; Kiartzis, S.J.; Bakirtzis, A.G.; Theocharis, J.B.; Tsagas, G.Electrotechnical Conference, 2000. MELECON 2000. 10th Mediterranean Volume 3, Issue , 29-31 May 2000 Page(s): 1097 - 1100 vol.3.

Tools

There are several tools available to process fuzzy rules once they have been extracted. This section will outline their form.

Check out The Scikit-fuzzy Documentation, Release 0.2

Scikit Fuzzy Example

Example (Ruoting Wang)

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

age = ctrl.Antecedent(np.arange(0, 100, 1), 'age')
```

Loading [Contrib]/a11y/accessibility-menu.js

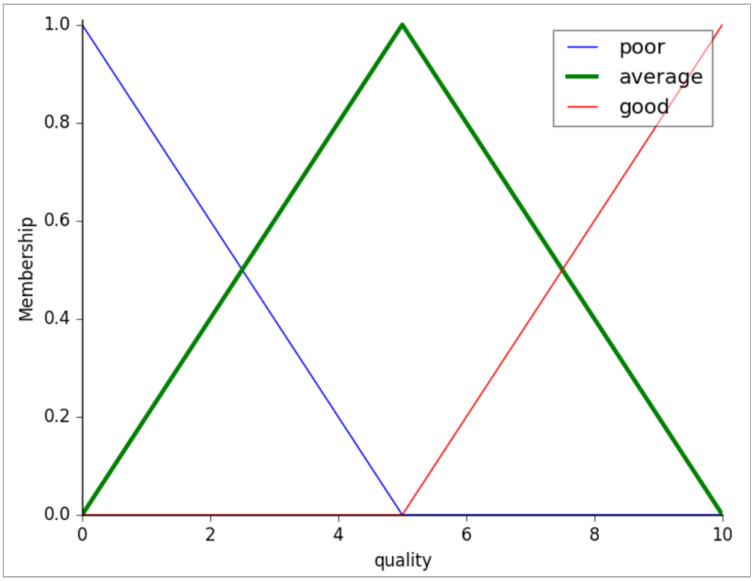
```
    ctrl.Consequent(np.arange(0, 95, 1), 'height')

# Fuzzy
age['low'] = fuzz.trimf(age.universe, [2, 4, 8])
height['small'] =
    fuzz.trimf(height.universe, [15, 30, 35])

age.view()
height.view()

# Define a rule
rule1 = ctrl.Rule(age['low'], height['small'])

control = ctrl.ControlSystem([rule1])
control_simulation =
    ctrl.ControlSystemSimulation(control)
control_simulation.input['age'] = 5
control_simulation.compute()
height.view(sim=control_simulation)
```

Scikit Fuzzy Example

```
# Based on
# https://pythonhosted.org/scikit-fuzzy/
auto_examples/plot_tipping_problem_newapi.html

import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Fuzzy variables quality, … defined
quality =
  ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
  # FLV quality has only 1 value

# Fuzzy variables service and tip are defined
service =
  ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
# Auto-membership function options 3, 5, or 7
quality.automf(3)   # on [0, 11] creates standard
                    # 'low', 'medium', and 'high'
service.automf(3)
```

Loading [Contrib]/a11y/accessibility-menu.js

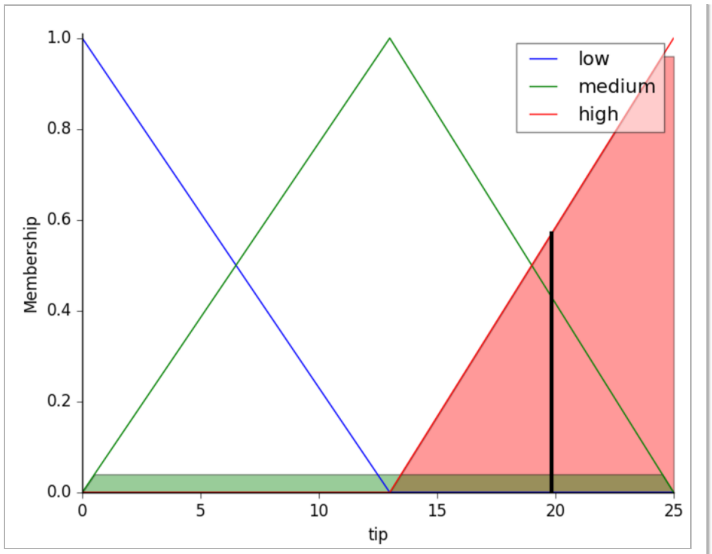
```
                    # on [0, 11] creates standard
                    # 'low', 'medium', and 'high'
```




```
# Custom fuzzy values for FLV tip
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

rule1 = ctrl.Rule(quality['poor'] |
        service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] |
        quality['good'], tip['high'])

# Control system consisting of these rules
tipping_ctrl
   = ctrl.ControlSystem([rule1, rule2, rule3])

# Ready to run
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
```

```
# Example
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8

# Calculate output
tipping.compute()
print(tipping.output['tip'])
tip.view(sim=tipping)
```

Loading [Contrib]/a11y/accessibility-menu.js



Source: # https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html

FuzzyJ API

`FuzzyRule` is a class as described, part of the `FuzzyJ` library.

```
public class FuzzyRule
extends java.lang.Object
```

A `FuzzyRule` holds three sets `FuzzyValues` for the antecedents, conclusions, and input values of a rule. A rule might be written as follows:

> if \(\text{antecedent}_1\) and \(\text{antecedent}_2\) and … \(\text{antecedent}_n\) then \(\text{conclusion}_1\) and \(\text{conclusion}_2\) and … \(\text{conclusion}_m\)

FuzzyJ Example 1

Recall that the values of the variables (antecedents and consequents—called “conclusions” here).

> Rule:
>
> **if** \(\text{antecedent}_1\) and \(\text{antecedent}_2\) and … \(\text{antecedent}_n\)
> **then** \(\text{consequent}_1\) and \(\text{consequent}_2\) and … \(\text{consequent}_m\)

Loading [Contrib]/a11y/accessibility-menu.js

Example:

if temperature is hot

then pressure is low or medium

Source: http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJDocs/FuzzyRule.html

FuzzyJ Example 2

The values of a fuzzy variable are fuzzy sets. In `FuzzyJ`, they are represented in shorthand. For example, `hot` is a value of the fuzzy variable `temp`. The fuzzy set `hot` is the trapezoid with corners (25, 0) and (35, 1) in the xy plane.

```
// some values used to describe the fuzzy
//terms in the temperature FuzzyVariable
double xHot[] = {25, 35};
double yHot[] = {0, 1};
double xCold[] = {5, 15};
double yCold[] = {1, 0};
// define our temperature FuzzyVariable
//with terms hot, cold, very hot and medium
FuzzyVariable temp = new
FuzzyVariable("temperature", 0, 100, "C");
temp.addTerm("hot", xHot, yHot, 2);
temp.addTerm("cold", xCold, yCold, 2);
temp.addTerm("veryHot", "very hot");
temp.addTerm("medium", "(not hot and (not cold))");
```

Source: http://rorchard.github.io/FuzzyJ/

FuzzyJ Example 3

`FuzzyJ` has classes that facilitate the definition of fuzzy sets in pre-formed shapes such as `ZFuzzySet` (a flattened z-shape).

```
// define our pressure FuzzyVariable
// with terms low, medium and high
FuzzyVariable pressure =

new FuzzyVariable("pressure", 0, 10, "kilo-pascals");
pressure.addTerm("low", new ZFuzzySet(2.0, 5.0));
pressure.addTerm("medium", new PIFuzzySet(5.0, 2.5));
pressure.addTerm("high", new SFuzzySet(5.0, 8.0));
```

Loading [Contrib]/a11y/accessibility-menu.js

Source: http://rorchard.github.io/FuzzyJ/

FuzzyJ Example 4

Given fuzzy linguistic variables, it becomes possible to create rules as in the code below.

```
// build a rule ---
FuzzyRule rule1 = new FuzzyRule();
FuzzyValue antecedentFval =
new FuzzyValue(temp, "hot");
FuzzyValue conclusionFval =
new FuzzyValue(pressure, "low or medium");
FuzzyValue inputFval =
new FuzzyValue(temp, "very medium");
rule1.addAntecedent(antecedentFval);
rule1.addConclusion(conclusionFval);
rule1.addInput(inputFval);
```

Source: http://rorchard.github.io/FuzzyJ/

FuzzyJ Example 5

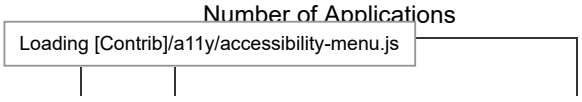
`FuzzyJ` includes tools to inspect the operation of individual rules.

```
// execute this simple rule with a single antecedent and
// a single consequent using default rule executor --
// MamdaniMinMaxMinRuleExecutor
FuzzyValueVector fvv = rule1.execute();
// show the results using the plotting methods for FuzzyValues
FuzzyValue fvals[] = new FuzzyValue[2];
fvals[0] = antecedentFval;
fvals[1] = inputFval;
System.out.println(FuzzyValue.plotFuzzyValues("*+", 0, 50, fvals));
System.out.println(fval2.plotFuzzyValue("*", 0, 10));
System.out.println(fvv.fuzzyValueAt(0).plotFuzzyValue("*", 0, 10));
```

Source: http://rorchard.github.io/FuzzyJ/

Conclusion

The number of applications worldwide became essentially uncountable in 2007.



Loading [Contrib]/a11y/accessibility-menu.js

Year	#
1985	8
1987	15
1988	50
1989	100
1990	150
1991	300
1992	800
1993	1500
2007	\(\infty\)

Example of Recent Application

The ease with which fuzzy rules can be formulated continues to encourage their use and formulation from data. The following shows a recent application.

- Evaluating the Service Quality of Hospital Using Topsis with Interval Type-2 Fuzzy Sets
- 2017 International Conference on Fuzzy Theory and Its Applications

Impact

The impact of fuzzy applications has waxed and waned over the past decade.

- Apps require fewer specifics from users – adjust automatically with fuzzy logic
  - washing machines
  - cameras
  - transportation
- Engineering of more inexact applications possible
  - automated car steering and/or warning
  - applications to safety

Summary

When fuzzy rules can be obtained and used, their operation is easy to understand.

- Introduce fuzzy linguistic variables
- Define fuzzy rules
- Use fuzzy expert systems where possible

Loading [Contrib]/a11y/accessibility-menu.js

Loading [Contrib]/a11y/accessibility-menu.js