Università degli Studi di Bari
Dipartimento di Informatica

LACAM
Machine Learning

CLIPS: Representing uncertainty

Antonio Vergari

May 27, 2015

Uncertainty (like in MYCIN)

CLIPS inference engine does not provide any mean to deal with uncertain facts or uncertain pattern matching. However, once one is able to project rules and matching ont an intermediate representation level, uncertainty factors can be introduced at different layers.

In MYCIN, ***certainty factors*** (**CF**) could be associated to facts in the knowlede base, representing the confidence of observing them (positive or negative[1]). Observing two identical facts with different CFs modifies the WM by introducing just one fact with a newly computed CF as follows:

$$CF(X,Y) = \begin{cases} X + Y - XY, & \text{if } X, Y > 0 \\ X + Y + XY, & \text{if } X, Y < 0 \\ \frac{X+Y}{1-\min\{|X|,|Y|\}}. & \text{otherwise} \end{cases}$$

[1] Originally they are weights in $\{-1, 1\}$. In the following wine example we will consider them normalized in $[0, 100]$

Uncertainty II

Intuitively, seeing more instances of a fact with positive CFs would increase the confidence about them, while negative CFs will lower it.

Moreover, if we consider patterns P_1 and P_2 in the LHS of an uncertain rule, we can derive the combined CF for the rule in this way:

$$CF(P_1 \text{ and } P_2) = \min\{CF(P_1), CF(P_2)\}$$

$$CF(P_1 \text{ or } P_2) = \max\{CF(P_1), CF(P_2)\}$$

$$CF(\text{not } P) = -CF(P)$$

If a rule with an uncertain pattern matches an uncertain fact, the resulting CF will be the product of the two CFs (resulting in a lower confidence).
If all CF are the max possible values (1), note how we fall into the certain pattern matching case.

Wine suggester

Back to the wine suggester toy problem, suppose that now the task is to suggest the user the name of a wine according to some desiderd properties: *color*, *body*, *sweetness*. Instead of suggesting only one possible wine, the ES will output a list of wines ordered by their CFs.

```
Do you generally prefer dry, medium, or sweet wines? dry
Do you generally prefer red or white wines? white
Do you generally prefer light, medium, or full bodied wines? light
Is the flavor of the meal delicate, average, or strong? delicate
Does the meal have a sauce on it? yes
Is the sauce for the meal spicy, sweet, cream, or tomato? tomato
Is the main component of the meal meat, fish, or poultry? meat
WINE                          CERTAINTY
_____________________________________
Valpolicella                  100%
Zinfandel                     64%
Cabernet—Sauvignon            40%
Soave                         40%
Chablis                       40%
```

Extending attributes and rules

Derivable facts can still be represented as attribute-value pairs, with an added certainty field as a float number (maxed at 100.0):

```
(deftemplate MAIN::attribute
    (slot name)
    (slot value)
    (slot certainty (default 100.0)))
```

We can still use the multislot representation for rules as facts, introducing the float property certainty even here to express the confidence of the rule

```
(deftemplate RULES::rule
    (slot certainty (default 100.0))
    (multislot if)
    (multislot then))
```

Wine suggester: example rules

Here are some uncertain rules (they start with a certainty of 100):

```
(rule (if has-sauce is yes and  sauce is spicy)
      (then best-body is full))

(rule (if tastiness is delicate)
      (then best-body is light))

(rule (if tastiness is average)
      (then best-body is light with certainty 30 and
            best-body is medium with certainty 60 and
            best-body is full with certainty 30))

(rule (if tastiness is strong)
      (then best-body is medium with certainty 40 and
            best-body is full with certainty 80))

(rule (if has-sauce is yes and sauce is cream)
      (then best-body is medium with certainty 40 and
            best-body is full with certainty 60))
```

Outline strategy

I. Ask the users about the wine and meal properties. Assert those facts with a 100% certainty

II. Execute a forward chaining step by using uncertain wine rules to assert (possibly) uncertain facts about the derived desired properties

III. Combine uncertain attributes about the same facts according to MYCIN rules

IV. Generate the wine list according to the derived attributes

V. Sort the generated list against the certainty factors

In this scenario one may want more identical facts to be asserted in the WM:

```
(defrule MAIN::start
    (declare (salience 10000))
    =>
    (set-fact-duplication TRUE)
    (focus QUESTIONS CHOOSE-QUALITIES WINES PRINT-RESULTS))
```

Questioning I

Askable questions are represented as templated facts whose slots keep track of valid answers as well as which other questions should be asked first (precursor).

```
(deftemplate QUESTIONS::question
    (slot attribute (default ?NONE))
    (slot the-question (default ?NONE))
    (multislot valid-answers (default ?NONE))
    (slot already-asked (default FALSE))
    (multislot precursors (default ?DERIVE)))
```

For instance, one cannot ask this second question if the first has not already been asked, and its answer was poultry:

```
(deffacts WINE-QUESTIONS::question-attributes
    (question (attribute main-component)
              (the-question "Is the main dish meat, fish, or poultry? ")
              (valid-answers meat fish poultry unknown))
    (question (attribute has-turkey)
              (precursors main-component is poultry)
              (the-question "Does the meal have turkey in it? ")
              (valid-answers yes no unknown)))
```

Questioning II

A question can be asked only when its **precursor** slot is empty

```
(defrule QUESTIONS::ask-a-question
   ?f <- (question (already-asked FALSE)
                   (precursors)
                   (the-question ?the-question)
                   (attribute ?the-attribute)
                   (valid-answers $?valid-answers))
   =>
   (modify ?f (already-asked TRUE))
   (assert (attribute (name ?the-attribute)
                      (value (ask-question ?the-question ?valid-answers)))))
```

We make it empty by removing matching patterns:

```
(defrule QUESTIONS::precursor-is-satisfied
    ?f <- (question (already-asked FALSE) (precursors ?name is ?value $?rest))
    (attribute (name ?name) (value ?value))
    =>
    (if (eq (nth 1 ?rest) and) then (modify ?f (precursors (rest$ ?rest)))
                               else (modify ?f (precursors ?rest)))))
```

Uncertain forward chaining I

Our modified forward chaining inference will proceed by selecting rules whose LHS matches an asserted attribute. If such an attribute exists, then the corresponding pattern is removed from the LHS[2]

```
(defrule RULES::remove-is-condition-when-satisfied
    ?f <- (rule (certainty ?c1)
                (if ?attribute is ?value $?rest))
    (attribute (name ?attribute)
               (value ?value)
               (certainty ?c2))
    =>
    (modify ?f (certainty (min ?c1 ?c2)) (if ?rest)))
```

and the global CF of the rule is updated to be the min of the current CF and the removed pattern CF (since it is the derivation for the CF of a conjunction of patterns).

---

[2] Like in the backward chaining setting, we use the WM as a blackboard storing only the rules and their parts that we can use again.

Uncertain forward chaining II

Again, when a rule fact in our WM gets an empty LHS (if multislot), then we can remove it and activate its RHS by asserting the corresponding attributes:

```
(defrule RULES::perform-rule-consequent-with-certainty
    ?f <- (rule (certainty ?c1)
                (if)
                (then ?attribute is ?value with certainty ?c2 $?rest))
    =>
    (modify ?f (then ?rest))
    (assert (attribute (name ?attribute)
                       (value ?value)
                       (certainty (/ (* ?c1 ?c2) 100)))))
```

In this case the CF of the attribute as fact to be asserted is equal to the product of the global CF and the RHS pattern CF.
Once all rules have been ``fired'', then the WM would contain the attributes from the user answers and those derived from the forward chaining inference step.

Combining uncertain facts

Different attribute facts representing the same piece of information are now being merged into single facts whose CF is computed as the in MYCIN:

```
(defrule MAIN::combine—certainties
    (declare (salience 100)
    (auto—focus TRUE))
    ?rem1 <— (attribute (name ?rel) (value ?val) (certainty ?per1))
    ?rem2 <— (attribute (name ?rel) (value ?val) (certainty ?per2))
    (test (neq ?rem1 ?rem2)) ;; why is this necessary?
    =>
    (retract ?rem1)
    (modify ?rem2 (certainty (/ (— (* 100 (+ ?per1 ?per2))
                                   (* ?per1 ?per2)) 100))))
```

Consider that in this case we are coping with only positive CFs.

Generating uncertain suggestions I

At this point we have to generate the results in terms of uncertainty for each wine. Suppose that we have such a KB about wines, described in terms of color, body and sweetness:

```
(deffacts WINES::the-wine-list
    (wine (name Gamay) (color red) (body medium)
          (sweetness medium sweet))
    (wine (name Chablis) (color white) (body light) (sweetness dry))
    (wine (name Sauvignon-Blanc) (color white) (body medium)
          (sweetness dry))
    (wine (name Valpolicella) (color red) (body light))
    (wine (name Cabernet-Sauvignon) (color red)
          (sweetness dry medium))
    (wine (name Zinfandel) (color red) (sweetness dry medium))
    (wine (name Pinot-Noir) (color red) (body medium)
          (sweetness medium))
    (wine (name Burgundy) (color red) (body full))
    (wine (name Zinfandel) (color red) (sweetness dry medium)))
```

Generating uncertain suggestions II

Now we can match them against our attributes in the WM:

```
(defrule WINES::generate-wines
    (wine (name ?name)
          (color $? ?c $?)
          (body $? ?b $?)
          (sweetness $? ?s $?))
    (attribute (name best-color) (value ?c) (certainty ?certainty-1))
    (attribute (name best-body) (value ?b) (certainty ?certainty-2))
    (attribute (name best-sweetness) (value ?s) (certainty ?certainty-3))
    =>
    (assert (attribute (name wine) (value ?name)
                       (certainty (min ?certainty-1
                                       ?certainty-2
                                       ?certainty-3)))))
```

Computing the resulting CFs as min values of the CFs, since we are dealing with a conjunction of patterns again.

Note that if more attributes facts about a wine are derived, then the combine rule will get activated again.

Ordering suggestions

The ordering procedure is composed by a rule getting (and retracting) the wine fact with highest CF in the WM:

```
(defrule PRINT-RESULTS::print-wine
    ?rem <- (attribute (name wine) (value ?name) (certainty ?per))
    (not (attribute (name wine) (certainty ?per1&:(> ?per1 ?per))))
    =>
    (retract ?rem)
    (format t " %-24s %2d%%%n" ?name ?per))
```

Wine suggestions getting a CF lower than a certain threshold (let's say 20), can be omitted:

```
(defrule PRINT-RESULTS::remove-poor-wine-choices ""
    ?rem <- (attribute (name wine) (certainty ?per&:(< ?per 20)))
    =>
    (retract ?rem))
```

Exercises

Add another filter to the output wine list, showing only the top K items, where K is a positive integer that shall be asked to the user.

Using this example as a basis, build an exert system for recommending some items in a domain you know. For instance, you can suggest the user the top movies he could like.
Explicit what are the main attributes for an item to be recommended (for the wine domain we had: color, body, sweetness).
Define a hierarchy of intermediate attributes of interest and to get their values write the proper questions with the proper precursor slot set.
Define a set of uncertain rules to be used for inference.

Devise a way to interrupt the question phase when you can already derive K suggested items with a threshold > 50.