**Introduction**

Weighted Threshold Operators are $n$-ary logical operators which compute a weighted sum of their arguments and verify whether it reaches a certain threshold. These operators have been extensively studied in the context of circuit complexity theory, and they are also known in the neural network community under the alternative name of perceptrons.

These operators can easily be added to most Description Logics, essentially at no extra computational cost, and they have natural applications in knowledge representation. Also, being fundamentally linear classifiers, they can be easily learned from data / extracted from richer ML models.

**Example 1: Course Offerings**

- Course A: 1 credit
- Course B: 1 credit
- Course C: 2 credits
- Course D: 2 credits
- A student must gain at least 3 credits.

\[ \text{Student} \sqsubseteq (C \sqcap D) \sqcup ((A \sqcup B) \sqcap (C \sqcup D)) \]

Semantically equivalent, but more human-readable and easier to update.

**Example 1 (continued):**

Suppose now that Course C gets assigned only 1 credit:

- Course A: 1 credit
- Course B: 1 credit
- Course C: 1 credit
- Course D: 2 credits
- A student must gain at least 3 credits.

\[ \text{Student} \sqsubseteq (A \sqcap B \sqcap C) \sqcup ((A \sqcup B \sqcup C) \sqcap D) \]

The threshold expression only requires a minimal change, but the expression written in terms of Boolean connectives needs to be changed completely!

**Definition and Complexity Results**

If $C_1, \ldots, C_n$ are concept expressions, $w_1, \ldots, w_n, t \in \mathbb{R}$, $\mathcal{W}(C_1 : w_1, \ldots, C_n : w_n)$ is a concept expression and

\[
(\mathcal{W}(C_1 : w_1, \ldots, C_n : w_n))^I = \{ d \in \Delta^I : \sum_{c \in d} w_c \geq t \}.
\]

- Does not increase expressive power (if usual Boolean connectives available);
- More human-understandable in many contexts;
- Easier to update;
- Easier to learn or extract (linear classifier).

**Theorem ([1]):**

Let $\mathcal{L}$ be a Description Logic that contains all Boolean connectives, let $\mathcal{K}$ be a $\mathcal{L}(\mathcal{W})$ knowledge base and let $\phi$ be a $\mathcal{L}(\mathcal{W})$ axiom (integer weights).

Then, the problem of whether $\mathcal{K} \models \phi$ can be reduced, with polynomial overhead, to the problem of whether $\mathcal{K}_L \models \phi_L$ for some $\mathcal{L}$ knowledge base $\mathcal{K}_L$ and some $\mathcal{L}$ axiom $\phi_L$.

**Learning Threshold Expressions**

Even very simple (<10 components) threshold expressions can express useful concepts, and can be learned efficiently from data!

As a simple example, we tried to learn the Molecular Function Gene Ontology [2] annotations of yeast proteins given their Cellular Component and Biological Process ones—and showed that even simple threshold expressions (TOOTH) can represent possible solutions that are roughly as accurate as the (much less easily interpretable) models learned by state-of-the-art learning algorithms.

But these expressions are much more readable, and can be easily added to an knowledge base / examine their implications given a knowledge base!

**Example 2: Florida Score Sheet**

- Possession of Cocaine: +16 “felony points”;
- Moderate injuries: +18 “felony points”;
- Failure to appear: +4 “felony points”;
- . . .

If total $\geq 44$, imprisonment is compulsory; otherwise, not so [3].

How to express **COMPULSORY_IMPRISONMENT** in a knowledge base? Certainly possible (e.g. Disjunctive Normal Form), but not very short or readable.

With threshold operators, much clearer and easier to update:

\[
\text{COMPULSORY_IMPRISONMENT} = \mathcal{W}^{11}(\text{COCAINE} : 16, \text{MODERATE_INJURIES} : 18, \ldots)
\]

**Conclusions**

Our results lend support to the feasibility of adding threshold connectives to knowledge representation languages. Such connectives can be added to the language of any DL that has all Boolean connectives without increasing the complexity of the corresponding inference problem, and thus reasoning services for any such DL $\mathcal{L}$ can be also used (after translation) also for the corresponding extension $\mathcal{L}(\mathcal{W})$.

Furthermore, as we showed in Section with a practical example over the Gene Ontology, even simple instances of perceptron connectives are expressive enough to represent complex notions in real use cases.

Future work on perceptron operators is currently being considered in a number of directions, including adding ‘counting capabilities’ to the language [4] as well as using perceptron operators to address compositionality and typicality effects in concept combination [5].

**References**


