Internship Offer (Master 1 or 2) — SDS Team — INSA CVL

Reasoning with probabilistic and contradictory information

Supervisors:

Sabine Frittella — sabine.frittella@insa-cvl.fr
Vincent Hugot — vincent.hugot@insa-cvl.fr

You can check that you have the latest version of this document and that the position is still available on https://tcs.vhugot.com/Offers.

---

0 Contents

1 Description of the internship

Context. Information can take various forms: *crisp*, *fuzzy*, *probabilistic*... whether we are asking a yes/no question, asking how tall someone is, or how likely it is to rain tomorrow. It can also be *incomplete* – when one does not have or cannot have access to some pieces of information – or *contradictory* – when one or many sources provide contradictory pieces of evidence. In addition, the information can also be *imprecise*: a measurement is given with a margin of error, one does not know the exact probability of it raining tomorrow, but one can provide an interval of probability with a certain degree of confidence...

Different mathematical frameworks enable us to reason with different kinds of pieces of information. For instance, *classical logics* are well suited for crisp non-contradictory information, while *paraconsistent logics* were developed to reason with contradictory information. Similarly, *imprecise probabilities* have been developed to formalise situations where a classical probability measure cannot be provided, for instance when one does not know the exact probability of an event, but can provide a lower bound on that probability.

This internship takes place within a project that aims to develop probability theory and probabilistic logics in a paraconsistent setting. We have already introduced paraconsistent probabilistic logics [3, 4] where we study probabilities and belief functions [2] to reason with contradictory and incomplete probabilistic information. Now, we wish to *extend this research to other notions of imprecise probabilities* such as *possibility functions* and *lower probabilities*. In addition, we intend to look at applications of this work by first *developing software to study this theory on real data* (for instance on this real, imprecise and uncertain birds dataset [5] generated through crowdsourcing specifically to run experiments on belief functions) and by *looking at potential applications* (one could start by looking at connections with [6] that uses belief functions together with the open-world assumptions to treat contradictory information from satellite imagery).

Goal of the internship. The goal of the internship is to develop software to study probabilities in a paraconsistent setting on real data. The intern will need to be comfortable with reasoning with functions over the powerset, manipulating basic notions about probabilities and classical propositional logic. In addition, we shall need to implement a large number of symbolic transformations on formulae; functional languages with structural pattern-matching, such as OCaml or Haskell, are ideal for such tasks, and candidates with this background will be greatly appreciated. Python (3.10+) is also suitable for rapid prototyping. We shall begin by implementing belief functions and standard combination rules such as Dempster Shafer's rule in a classical setting, then we shall generalise to the paraconsistent setting.

2 Some mathematical background

In this section we introduce the main mathematical notions that will be needed throughout the internship. Then we comment briefly on the generalisation of probability theory to the paraconsistent setting.

2.1 Probability measures

Consider a *random variable* X that takes its value in the finite set $\Omega = \{x_1, \ldots, x_n\}$. For instance, one can represent the throw of dice via a random variable X over $\Omega = \{1, 2, 3, 4, 5, 6\}$.

A *probability distribution* over Ω is a map $p : \Omega \to [0, 1]$ such that

$$\sum_{x_i \in \Omega} p(x_i) = 1.$$

$p(x_i)$ encodes the probability that the variable X takes the value x_i. If we consider our dice, we have $p(n) = \frac{1}{6}$ for $1 \leqslant n \leqslant 6$.

A *probability measure* over $\mathcal{P}(\Omega)$ is a map $\mu : \mathcal{P}(\Omega) \to [0, 1]$ such that

(1) $p(\Omega) = 1$ and

(2) $p(A \cup B) = p(A) + p(B)$ for all $A, B \subseteq \Omega$ such that $A \cap B = \varnothing$.[(a)]

One can show that any probability measure is monotone (that is, if $A \subseteq B$, then $p(A) \leqslant p(B)$).

2.2 Belief functions and the theory of evidence

Belief functions generalise the notion of probabilities. They enable us to treat some situations in which one cannot provide the specific probability of an event, but one can provide an interval.

A *mass function* over the set $\mathcal{P}(\Omega)$ is a function $\mathtt{m} : \mathcal{P}(\Omega) \to [0, 1]$ such that

$$\sum_{A \in \mathcal{P}(\Omega)} \mathtt{m}(A) = 1.$$

A *belief function* over $\mathcal{P}(\Omega)$ is a map $\mathtt{bel} : \mathcal{P}(\Omega) \to [0, 1]$ such that there is a mass function $\mathtt{m}$ over $\mathcal{P}(\Omega)$ such that : for every $A \subseteq \Omega$

$$\mathtt{bel}(A) = \sum_{B \subseteq A} \mathtt{m}(B).$$

[(a)] In this work, we will only consider probability measures over finite sets, therefore we can restrict ourselves to the axiom of finite additivity.

Mass functions are used to encode information. When the mass function $\mathtt{m} : \mathcal{P}(\Omega) \to [0,1]$ is non-zero only on singletons, its associated belief function $\mathtt{bel}$ is in fact a probability measure. Otherwise, $\mathtt{bel}(A)$ provides a lower bound on the probability that the random variable takes its value in A.

Since mass functions encode information and more specifically pieces of evidence, many rules have been proposed to combine mass functions, that is, to aggregate the information contained in many mass functions. One example is *Demspter-Shafer combination rule*.

Definition 2.1 (Dempster's combination rule over a powerset algebra)**.** Let $\mathtt{m}_1$ and $\mathtt{m}_2$ be two mass functions on a powerset algebra $\mathcal{P}(\Omega)$. Dempster's combination rule computes their aggregation $\mathtt{m}_1 \oplus \mathtt{m}_2$ as follows.

$$\mathtt{m}_1 \oplus \mathtt{m}_2 : \mathcal{P}(\Omega) \to [0,1] \tag{1}$$
$$X \mapsto \begin{cases} 0 & \text{if } X = \varnothing \\ \dfrac{\sum\{\mathtt{m}_1(X_1) \cdot \mathtt{m}_2(X_2) \mid X_1 \cap X_2 = X\}}{\sum\{\mathtt{m}_1(X_1) \cdot \mathtt{m}_2(X_2) \mid X_1 \cap X_2 \neq \varnothing\}} & \text{otherwise.} \end{cases}$$

First step of the internship. To code belief functions, Dempster-Shafer combination rule, and other similar combination rules from the literature.

The software will take as input a set Ω and a collection of mass functions over $\mathcal{P}(\Omega)$, each expressed as a collection of couples (A, x), where $A \subseteq \Omega$ and $x \in]0,1]$. From there, the Dempster-Shafer combination rule will be straightforward to implement.

This software will then be extended to classical logic and Belnap-Dunn logic, other combination rules, etc, depending on time and affinities. This will require implementing a small parser for propositional formulæ, a transformation into (non-redundant) Disjunctive Normal Form, etc.

2.2 A very non-technical presentation of classical propositional logic and Belnap-Dunn logic

Let $\mathtt{Prop}$ be a finite set of propositional variables. Formulas of classical propositional logic are defined by induction as follows:

$$\phi := p \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi$$

with $p \in \mathtt{Prop}$. When interpreted over a powerset algebra, the conjunction $\wedge$ can be interpreted as the intersection $\cap$, the disjunction $\vee$ as the union $\cup$, and the negation $\neg$ as the complement.

Classical propositional logic is used to reason about facts of the world. For instance, the statement *John lives in London* is either *true* or *false*. Belnap-Dunn logic 10 was introduced to

reason about the information available about a statement. For instance, one could have no information about where John lives, the information that he lives in London, the information that he does not live in London, or contradictory information about where he lives. To formalise the 4 possible situations regarding the available information about the statement *John lives in London*, one uses 4 truth values **T** (true), **F** (false), **N** (neither), **B** (both). These truth values are interpreted as follows:

- ◇ **T**: the available information supports the fact that John lives in London
- ◇ **F**: the available information supports the fact that John does not live in London
- ◇ **N**: there is no available information whether John lives in London
- ◇ **B**: there is contradictory information about whether John lives in London

Second step of the internship. Understand belief functions over Belnap-Dunn logic (these notions are introduced here [3] and will be explained by the supervisors), and take advantage of the fact that Belnap-Dunn logic can be encoded in classical logic to reuse the previous code to compute aggregation of mass functions in the setting of Belnap-Dunn logic.

Further work. Depending on the interests and the profile of the intern, many other tasks can be done, such as:

- ◇ Interface the software with the dataset available here [5], study the behaviour of different combination rules and compare them in the classical setting and in the paraconsistent setting.
- ◇ In steps 1 and 2, we will work over models, that is to say, computation will be done over powerset algebras like $\mathcal{P}(\Omega)$. It would be interesting to do a similar work but over De Morgan algebras that are used to interpret Belnap-Dunn logic.
- ◇ To develop a user-friendly interface to help the user interpret the result of the computations.
- ◇ ...

3 Practical information

3.1 Contacts & Application procedure:

Before applying, please check that you have the latest version of this document and that the position is still available: `https://tcs.vhugot.com/Offers`

Please submit your application by email to
Dr. Sabine Frittella (email: `sabine.frittella@insa-cvl.fr`) and
Dr. Vincent Hugot (email: `vincent.hugot@insa-cvl.fr`), including:

- your CV,
- your academic transcript for your Bachelor and Master's degrees (or equivalent),
- a short cover letter explaining your motivation for pursuing this specific internship.

3.2 Requirements

Your are:

- A student in Master 1 or Master 2 (or equivalent) in Computer Science, Mathematics, or a relevant discipline;
- Fluent in reading English;
- Fluent in writing and speaking English or French.

3.3 Additional Information

- Duration and stipend: the internship is for 4 to 6 months, with a stipend of approximately 600 euros per month.
- Location: Department of Computer Science at INSA Centre Val de Loire, Bourges, France.
  Low-rate university accommodation as well as a university restaurant are available.

4 References

(1) R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi. Reasoning About Knowledge. MIT press. 1995.

(2) G. Shafer, A Mathematical Theory of Evidence. Princeton University Press. 1976.

(3) M. Bílková, S. Frittella, D. Kozhemiachenko, O. Majer, S. Nazari. Reasoning with belief functions over Belnap–Dunn logic. Annals of Pure and Applied Logic. 2023. PDF

(4) M. Bílková, S. Frittella, D. Kozhemiachenko, O. Majer, K. Manoorkar: Describing and quantifying contradiction between pieces of evidence via Belnap-Dunn logic and Dempster-Shafer theory. ISIPTA 2023: 37-47. PDF

(5) C. Thierry, A. Hoarau, A. Martin, J-C. Dubois, Y. Le Gall: Real Bird Dataset with Imprecise and Uncertain Values. BELIEF 2022: 275-285. PDF

(6) E. Skau, C. Armstrong, D. P. Truong, D. Gerts, K. Sentz: Open world Dempster-Shafer using complementary sets. ISIPTA 2023: 438-449. PDF

(7) J. Marques-Silva. "Practical applications of boolean satisfiability." In 2008 9th International Workshop on Discrete Event Systems, pp. 74-80. IEEE, 2008. PDF

(8) Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," in IEEE Transactions on Computers, vol. C-35, no. 8, pp. 677-691, Aug. 1986. PDF

(9) A. Chandra, D. Kozen, and L. Stockmeyer. 1981. Alternation. J. ACM 28, 1 (Jan. 1981), 114–133. PDF

(10) N. Belnap, How a Computer Should Think, Springer, Cham, Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science), New Essays on Belnap-Dunn Logic, 418, 2019.