Removing Redundancies for Faster Inference

Arne Decadt ARNEDECADT@UGENT.BE
Foundations Lab for imprecise probabilities, Ghent University, Belgium

When an inference model is complex, it can also be slow and clunky. That's why we will look at a new technique for removing redundancies from a class of models, making it faster and easier to use. We use sets of strictly desirable gambles.

Let's consider a finite set $\mathcal{X}$ that encompasses all possible outcomes of an experiment. Gambles, which are mappings from $\mathcal{X}$ to a utility scale, form a real vector space denoted as $\mathbb{R}^{\mathcal{X}}$. We define $\mathbb{R}^{\mathcal{X}}_{\geq 0}$ as the set of gambles with non-negative components, 0 as the zero gamble, and $\mathcal{E}(A)$ as a positive hull operator $\mathcal{E}(A) := \left\{\sum_{k=1}^{n} \lambda_k f_k : \lambda_k > 0 \wedge f_k \in A \cup (\mathbb{R}^{\mathcal{X}}_{\geq 0} \setminus 0) \wedge n \in \mathbb{N}\right\}$ for any $A \in \mathbb{R}^{\mathcal{X}}$. If $0 \notin \mathcal{E}(A)$, we say that the assessment A is consistent, and $\mathcal{E}(A)$ represents its natural extension. In this context, the common approach for making inferences involves checking if a given gamble is present in the natural extension.

Now, let us examine the case where A is finite. The most straightforward method to achieve this is by verifying if a given gamble is greater than or equal to a positive linear combination of a finite number of gambles, which can be solved using linear programming techniques [3]. Suppose $\mathbf{A}$ represents the matrix composed by concatenating the gambles of A as column vectors, λ is an unknown vector with $|A|$ components and g the gamble we want to check for inclusion in the natural extension. In this scenario, we need to determine if the following program has a solution: *find* λ *subject to* $\mathbf{A}\lambda \leq g, \lambda \geq 0, \sum \lambda \geq 1$. Alternatively, for low dimensions $d = |\mathcal{X}|$, we can compute the credal set by finding the dual polytope, and then check whether the lower expectation is non-negative. This approach becomes more computationally expensive for higher dimensions as the number of extreme probability mass functions given k desirable gambles can be as high as $\binom{k - \lceil d/2 \rceil}{\lfloor d/2 \rfloor} + \binom{k - \lceil d/2 \rceil - 1}{\lfloor d/2 \rfloor - 1}$, which grows exponentially in d. Computer evidence with random polytopes suggests that for high dimension one attains this bound with high probability [2, p. 394-395].

Suppose we employ the direct approach and need to check multiple gambles for inclusion in the natural extension. Then it can be advantageous to remove redundant gambles from A and find another smaller gamble set A' such that $\mathcal{E}(A') = \mathcal{E}(A)$. This problem is essentially finding the extreme rays of $\mathcal{E}(A)$, for which algorithms are typically associated with the convex hull problem. Here the same division of approaches can be made. Some approaches, e.g. QHULL, use the dual polytope but have problems in high dimensions due to the large number of facets [1]. Therefore many libraries, such as CDDLIB and POLYHEDRA.JL simply check for every $a \in A$ if $a \in \mathcal{E}(A \setminus \{a\})$, resulting in $|A|$ linear programs.

For consistent sets of desirable gambles (where $0 \notin \mathcal{E}(A)$), I have discovered an improved method. Initially, we employ linear programming to find a hyperplane that intersects all rays of the set of desirable gambles. This hyperplane can be defined as all gambles g for which the inner product $\lambda^T g = 1$, where λ is the solution to the linear program *find* λ *subject to* $\mathbf{A}^T \lambda \geq 1, \lambda \geq 1$. Then on this hyperplane we can remove one of the coordinates and find the convex hull using $|A|$ linear programs in this lower-dimensional space. The last coordinate can be retrieved with $\lambda^T g = 1$.

In the table I have put the median times in milliseconds of some tests, which always involved 40 different consistent assessments A, each containing 100 gambles. In a two-dimensional case, this method was approximately 100 times faster than the naive method, although the two-dimensional scenario is not particularly interesting since the lower prevision can be easily computed. The brute-force approach wins slightly until approximately dimension 30 when the new method starts winning again. Furthermore, I noted that as the imprecision of the 'real' model increased, the improvement diminished.

Dim.	New	Naive
2	0.437	35.433
30	106.862	102.429
50	223.020	260.479

References

[1] David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 20–28, 1995. doi:10.1145/220279.

[2] Martin Henk, Jürgen Richter-Gebert, and Günter M Ziegler. Basic properties of convex polytopes. In *Handbook of discrete and computational geometry*, pages 383–413. Chapman and Hall/CRC, 2017. doi:10.1201/9781315119601.

[3] Erik Quaeghebeur. The conestrip algorithm. In *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, pages 45–54. Springer, 2013. doi:10.1007/978-3-642-33042-1_2.