Identifying Norms from Observation Using MCMC Sampling

Stephen Cranefield[1*] and **Ashish Dhiman**[2]
[1]Department of Information Science, University of Otago, Dunedin, New Zealand
[2]Department of Aerospace Engineering, Indian Institute of Technology, Kharagpur, India
stephen.cranefield@otago.ac.nz, ashish1610dhiman@gmail.com

Abstract

To promote efficient interactions in dynamic and multi-agent systems, there is much interest in techniques that allow agents to represent and reason about social norms that govern agent interactions. Much of this work assumes that norms are provided to agents, but some work has investigated how agents can identify the norms present in a society through observation and experience. However, the norm-identification techniques proposed in the literature often depend on a very specific and domain-specific representation of norms, or require that the possible norms can be enumerated in advance. This paper investigates the problem of identifying norm candidates from a normative language expressed as a probabilistic context-free grammar, using Markov Chain Monte Carlo (MCMC) search. We apply our technique to a simulated robot manipulator task and show that it allows effective identification of norms from observation.

1 Introduction

The presence of social norms is a significant factor in the behaviour exhibited in human societies. While norms govern the behaviour of individuals, they have the emergent effect of making the society as a whole more predictable and effective. In multi-agent systems (MAS), software agents are often modelled with traits borrowed from humans, and norms are no exception. Normative agents [Andrighetto *et al.*, 2013], like people, have autonomy, but are equipped with computational means to reason about norms and weigh up the normative consequences of different courses of action (e.g. the risk of being sanctioned). When norms are known to be followed to a sufficient degree in the MAS, they can also help predict the behaviour of other agents. Thus, when norms and norm-aware agents are present, a multi-agent system can be expected to operate more efficiently.

This leads to the question of how agents come to know the norms that govern them. Early work on agent societies mostly assumed that codification of norms would be done by humans, as a step in the design of a society of software agents [Dellarocas and Klein, 2000]. The problem of *norm identification* considers open multi-agent systems in which no central authority imposes or proposes norms, and considers how agents can identify norms that are already prevalent in a society or group, or are exhibited by a role model, through their own experience and/or observations of others. As agents learn norms from each other, norms can spread and evolve in the society [Savarimuthu and Cranefield, 2011].

A variety of techniques have been applied to this problem, including association rule-mining [Savarimuthu *et al.*, 2010; Savarimuthu *et al.*, 2013], plan recognition [Oren and Meneguzzi, 2013], Bayesian learning [Cranefield *et al.*, 2016], Dempster-Shafer Theory [Sarathy *et al.*, 2017] and inductive logic programming [Tan *et al.*, 2019]. Scenarios adressed in these studies include socially disapproved behaviour such as littering and failure to tip in a restaurant, normative constraints governing movement on a transport network, appropriate actions in a library vs. a boardroom, and a social robotics application.

Much of this work proposes mechanisms that are specific to a particular representation of norms and the sources of information available in the targeted application domain. However, Bayesian and Dempster-Shafer learning offer promise as generic approaches that can be adapted to different application domains, do not *a priori* limit the expressiveness of norms, and do not impose restrictions such as reinforcement learning's usual Markov assumption.

Norm identification differs from the problem of norm synthesis [Campos *et al.*, 2010; Morales *et al.*, 2013; Morales *et al.*, 2015], which considers the automated adaptation or design of norms for a multi-agent system, in order to suppress undesired states. The approach aims to provide trusted advisor agents or a central authority to monitor the MAS, detect when norms should be changed, generate improved ones, and broadcast these to the agents. It thus makes strong assumptions about the MAS structure, which are not realistic in open agent societies with a peer-to-peer architecture. Norm identification is also distinct from the use of multi-agent reinforcement learning to learn coordinated strategies for social dilemmas [Sen and Airiau, 2007; Wang *et al.*, 2019; Zhang *et al.*, 2019; Chaudhuri *et al.*, 2019], as it aims to identify symbolic representations for norms and is performed prior to any attempt to validate the norms through the learner's own behaviour.

*Contact author

A limitation of prior work on Bayesian norm identification is that it requires a finite set of candidate norms to be determined in advance [Cranefield *et al.*, 2016]. This limits the expressiveness of the norm language, as recursively defined terms may not appear unless a finite depth is imposed. In this paper, we remove this restriction by adapting a technique used in program synthesis, i.e. the generation of computer programs that satisfy a given specification [Gulwani *et al.*, 2017], to norm identification. In particular, our work is based on the Bayesian program synthesis method of Saad et al. [2019], which was developed to synthesize probabilistic programs that model observed data sets. The method modifies Markov Chain Monte Carlo search (widely used with numeric data) to produce a posterior probability distribution over a language generated by a probabilistic grammar.

This paper makes the following contributions. First, we recast the method of Saad et al. as an application of the well-known Metropolis-Hastings MCMC algorithm. We then propose a modification of the algorithm to ensure that norms that are relevant to an observed task are favoured over irrelevant ones. As convergence was not addressed in the prior work, we propose the use of a vector representation of trees to allow the standard $\hat{R}$ convergence statistic to be applied to MCMC search over symbolic expressions. We then evaluate the approach through norm identification experiments in a simulated robot task, and find that it correctly identifies the real norm underlying the simulated behaviour. Therefore, we overcome the prior limitation of Bayesian norm identification by providing a generic method that can learn norms from a countably infinite language. Our code and supplementary material can be found online.[1]

2 MCMC Sampling

Markov Chain Monte Carlo (MCMC) methods use a random search process to sample from a probability distribution. They are commonly used in Bayesian inference to generate the posterior distribution of a model parameter θ given an observed set obs of instances of model variables, i.e. $p(\theta|obs)$. Algorithm 1 shows the Metropolis-Hastings algorithm [Gelman *et al.*, 2013] that is used in this work. This creates a "chain" of samples $\{\theta_i : 1 \leq i \leq n\}$ such that the distribution of sample values converges to $p(\theta|obs)$. An initial value θ^0 is sampled from a selected *starting distribution* $p_0(\theta)$, such that $p(\theta^0|obs) > 0$ (line 3). The chain is then generated by an iterative process: in each iteration i, a proposed new value θ^* is sampled from a *jumping distribution* $J(\theta^*|\theta^{i-1})$ (line 5). An acceptance rate r is then calculated (line 6) in terms of the prior $p(\theta)$, likelihood $p(obs|\theta)$ and the jumping distribution. The proposal θ^* is chosen to be the next element of the chain with probability $min(r, 1)$; otherwise $\theta^i = \theta^{i-1}$ (line 7).

In practice, after the chain is generated, an initial "warm-up" segment is discarded to reduce the influence of the starting value on the generated sample. The efficiency of the MCMC search is greatly affected by the choice of the starting and jumping distributions.

[1] https://git.io/JsVuF

Algorithm 1 The Metropolis-Hastings algorithm

1: **procedure** METROPOLIS-HASTINGS(obs, n)
2: *obs: observed data; n: num. samples desired*
3: Sample $\theta^0 \sim p_0(\theta)$ such that $p(\theta^0|obs) > 0$
4: **for** $i = 1, \cdots, n$ **do**
5: Sample $\theta^* \sim J(\theta^*|\theta^{i-1})$
6: $r = \dfrac{p(\theta^*|obs)/J(\theta^*|\theta^{i-1})}{p(\theta^{i-1}|obs)/J(\theta^{i-1}|\theta^*)}$
7: $\theta^i = \begin{cases} \theta^* & \text{with probability } min(r,1) \\ \theta^{i-1} & \text{otherwise} \end{cases}$
8: **end for**
9: Return $\langle \theta^1, ..., \theta^n \rangle$
10: **end procedure**

3 MCMC Over a Probabilistic Grammar

MCMC search is usually applied to a numeric domain. In contrast, Saad et al. [2019] used MCMC search to sample from a space of symbolic expressions (programs that are generated by a grammar) given a set of data that the programs should generate. They defined a new type of probabilistic context-free grammar (PCFG): Tagged Probabilistic Context-Free Grammars with Random Symbols. In these grammars, each production rule must generate an s-expression beginning with a "phrase tag" unique to that production. There is a designated start symbol selected from the grammar's non-terminal symbols. A probability distribution is defined over the production rules for each non-terminal symbol, and for "non-recursive" rules (those with no non-terminal symbols following the phrase tag), an s-expression is produced containing the tag followed by a terminal symbol sampled from an associated probability distribution.

Figure 1 shows the grammar used in our norm-identification experiment described in Section 5. We annotate the '`::=`' and '`|`' symbols with the probabilities of the associated productions. These annotations are suppressed where the probability is 1. The meanings of the expressions generated by this grammar are discussed later.

Saad et al. define an MCMC search over expressions, which (although not presented that way) is a Metropolis-Hastings algorithm with the following choices. The grammar's probabilities on production rules and non-terminal symbol values define a prior distribution over expressions. This is used as the MCMC starting distribution, i.e. a chain's start value is generated from the start symbol using weighted random choice to select production rules and terminal symbols. Ensuring that the posterior probability of this value is non-zero (line 3) reduces to checking that the likelihood is non-zero, as the prior has already been used to generate the initial value and is therefore non-zero.

The jumping distribution is defined as follows:

- A node n in the current expression θ is chosen by uniform random selection, with probability $1/|\theta|$ where $|\theta|$ is the size of θ.
- The non-terminal symbol that was used to generate the selected node is determined.
- A new sub-expression is randomly generated in a similar

$$
\begin{aligned}
\text{NORMS} &\overset{0.5}{::=} \texttt{(no-norm t)} \\
&\overset{0.25}{\mid} \texttt{(norm NORM1)} \\
&\overset{0.25}{\mid} \texttt{(norms NORM1 NORM2)} \\
\text{NORM1} &\overset{0.5}{::=} \texttt{(obl COND ZONE)} \\
&\overset{0.5}{\mid} \texttt{(pro ACT COL SHAPE ZONE)} \\
\text{NORM2} &::= \texttt{(per ACT COL SHAPE PERZONE)} \\
\text{COND} &\overset{0.\dot{3}}{::=} \texttt{(moved COL SHAPE ZONE COND)} \\
&\overset{0.\dot{6}}{\mid} \texttt{(next-move COL SHAPE)} \\
\text{PERZONE} &::= \texttt{(per-zone } pz\texttt{)} \\
\text{ZONE} &::= \texttt{(zone } z\texttt{)} \\
\text{ACT} &::= \texttt{(action } a\texttt{)} \\
\text{COL} &::= \texttt{(colour } c\texttt{)} \\
\text{SHAPE} &::= \texttt{(shape } s\texttt{)}
\end{aligned}
$$

where

$$
\begin{gathered}
P(t = \texttt{true}) = 1 \\
P(c \in \{\texttt{r}, \texttt{g}, \texttt{b}\}) = \tfrac{1}{6} \quad P(c = \texttt{any}) = \tfrac{1}{2} \\
P(s \in \{\texttt{triangle}, \texttt{square}, \texttt{circle}\}) = \tfrac{1}{6} \\
P(s = \texttt{any}) = \tfrac{1}{2} \quad P(a = \texttt{putdown}) = 1 \\
P(z \in \{\texttt{1}, \texttt{2}, \texttt{3}\}) = \tfrac{1}{3} \\
P(pz \in \{\texttt{1}, \texttt{2}, \texttt{3}\}) = \tfrac{1}{6} \quad P(pz = \texttt{any}) = \tfrac{1}{2}
\end{gathered}
$$

Figure 1: A PCFG grammar for a language of norms

way to the initial expression, but starting with the non-terminal identified in the previous step.

- θ is modified to create θ^* by replacing the subtree at n with the new sub-expression.

Having chosen the jumping distribution, it is necessary to find a computationally efficient formula for the acceptance rate r. Saad et al. [2019] propose a computational formula for r, and prove its correctness. In the supplementary material, we show how this formula can be derived from line 6 in the Metropolis-Hastings algorithm (Algorithm 1).

As we are applying MCMC-based norm identification to a scenario in which the performance of a specific task is being observed (see Section 5), we are specifically interested in norm expressions θ that are *relevant* to the task, i.e. those for which $p(obs|\theta) > p(obs|no\text{-}norm)$, where $no\text{-}norm$ is the assumption that there is no norm. Expressions that do not satisfy this condition carry no explanatory power. Formally, we are interested in the posterior with an extra task condition t, i.e. $p(\theta|obs, t)$.

We take the observed task executions as proxies for knowedge of the task, and define an *irrelevance* relation:

$$\text{irrel}(\theta, obs) \equiv p(obs|\theta) \leq p(obs|no\text{-}norm)$$

We modify the Saad et al. definition of r as follows for a selected α, $0 < \alpha < 1$, where $\mathbb{1}$ is the indicator function that maps a proposition to 0 or 1. We have added the two powers of α. Essentially, this down-scales the prior probabilities for irrelevant expressions.

$$r = \frac{|\theta^{i-1}|\, \alpha^{\mathbb{1}(\text{irrel}(\theta^*, obs))}\, p(obs|\theta^*)}{|\theta^*|\, \alpha^{\mathbb{1}(\text{irrel}(\theta^{i-1}, obs))}\, p(obs|\theta^{i-1})} \tag{1}$$

It remains to define the likelihood $p(obs|\theta)$ of the observed data given an expression generated by the grammar. This is domain dependent. In our case of identifying norms, it is the likelihood of a sequence of observed agent actions given a norm expression. We discuss this in the context of our example norm-identification scenario in Section 5.

4 Convergence Testing

When using MCMC search, it is good practice to produce multiple chains, which can be examined visually in a trace plot [Lee and Wagenmakers, 2014] or statistically [Gelman *et al.*, 2013] for evidence that the chains have converged to the same, stationary, distribution. Techniques for doing this with numerical data are well established. However, Saad et al. [2019] do not address convergence testing for their MCMC search through expressions in a language generated by a grammar. The Gelman-Rubin convergence diagnostic [Gelman *et al.*, 2013] involves generating a number of independent chains from separate runs of an MCMC sampling algorithm, starting from "overdispersed" starting values. The first half of each chain is discarded to reduce the influence of the starting values. Each chain is then split into two. A comparison of the between-chain and within-chain variances results in a statistic $\hat{R}$ that should decrease to 1 as the chain size increases to infinity. Thus, $\hat{R}$ is computed for increasingly long initial subsequences of the chains. If the value is high, then there is reason to believe that further increasing the chain length is worthwhile [Gelman *et al.*, 2013].

As this diagnostic test is based on the variances of sample sets, it is necessary to have a distance metric defined over the sample space (symbolic expressions in our case). We adopt the inner product on trees that underlies the tree kernel of Collins and Duffy [2002]. This is based on a vector representation $\mathbf{h}(T) = \langle h_1(T), h_2(T), \cdots h_n(T) \rangle$ where the indices of the vector $\mathbf{h}$ are all the tree fragments appearing in T, and each element $h_i(T)$ is the number of times the tree fragment i appears in T. We then define the distance between two trees as $dist(T_1, T_2) = \sqrt{\mathbf{h}(T_1 - T_2) \cdot \mathbf{h}(T_1 - T_2)}$. In practice, we represent the tree vector for T as a Python Counter object initialised with a list of all subtrees of T, and only use complete (non-truncated) subtrees as our tree fragments.

We also use this vector representation of expressions to generate overdispersed starting expressions for the chains. 10 times the required number of starting expressions are randomly generated, and the distances between each expression's vector and the mean vector are calculated. The candidate starts are then sorted by descending order of distance, and the top ten are chosen as the start elements of the chains.

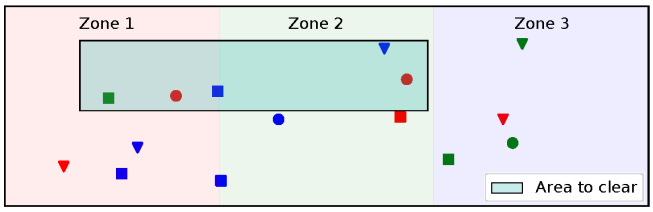

Figure 2: An example task to clear a region of the workspace (best viewed in colour)

```
(norms
  (obl (moved (colour 'any') (shape 'circle') (zone '2')
         (moved (colour 'g') (shape 'square') (zone '2')
           (next-move (colour 'b') (shape 'triangle'))))
       (zone '3'))
  (per (action 'putdown')
       (colour 'g') (shape 'triangle') (per-zone '1')))
```

Figure 3: An expression generated from the grammar in Figure 1

5 Example Domain

In this section we describe the simulated scenario used to evaluate our norm identification approach. Loosely based on the scenario of Tan et al. [2019], we consider a robot arm given a task to clear a user-specified region of a workspace (which may be shared with a human). Figure 2 shows an example state of the workspace and task. The workspace is divided into three zones (indicated by colours in the figure), numbered 1 to 3 from left to right. The workspace contains blocks with varying shapes (square, circular or triangular prisms) and colours (red, blue and green). To perform an instance of the task, for each block b in the region, the robot must perform the action $pickup(b)$ followed by $putdown(b, r)$ for some zone r. The order in which blocks are moved is not specified by the task, and nor are the new locations of the moved blocks. However, these may be constrained by a norm.

An agent observing one or more robots attempts to identify the norm (if any) governing the task performance, under the assumption that the norm is generated by the grammar in Figure 1. There is a true norm that the robots are aware of, and that they (mostly) comply with. The proportion of non-normative task executions is an experimental parameter, which we assume the observer knows (or has estimated accurately).

Each task execution is a sequence of pickup-putdown action pairs: one pair for each block to be moved. The observer applies the Metropolis-Hastings algorithm given the observed task executions to produce a sample of expressions. The frequency distribution of expressions in this sample approximates the posterior distribution over the norm language, provided that the MCMC chains have converged to a stationary distribution (see Section 4). It then extracts the norm expression with the highest posterior probability, or can use the posterior sample for posterior predictive inference.

5.1 The Grammar for Norms

The grammar in Figure 1 generates expressions that contain one or two norms, or an expression meaning that there is no norm. In the one-norm case, expressions may contain an obligation or a prohibition, and the two-norm case allows one of these to be combined with a permission. We use the *strong* notion of permission [Royakkers, 1997], where permissions represent exceptions to obligations or prohibitions.

Figure 3 shows an example expression containing an obligation and a permission. The obligation contains a nested sequence of three conditions.

The permission states that it is permitted to put green triangles down in zone 1. In this case, the permission is superfluous. As the obligation does not apply to green triangles, there is no restriction for the permission to override, and thus it is logically equivalent to a single-norm expression containing only the obligation. This non-redundant version of the expression has a higher prior probability than the redundant version, because the latter requires applying more production rules, each with their own probabilities to be factored into the prior. Given that the likelihood of any observed task execution will be the same given either expression as the true norm, the posterior probability of the non-redundant version will be higher, and it should be preferred by any Bayesian norm identification mechanism.

If the grammar had generated a prohibition for the first norm, rather than an obligation, it would have the form `(pro (action` a`) (colour` c`) (shape` s`) (zone` pz`))`. This states (unconditionally[2]) that it is prohibited to apply action a to a block of colour c, shape s, and zone pz (for 'prohibition zone'). The bottom of Figure 1 shows the possible values for the variables a, c, s and pz, with their probabilities. The specified colour, shape and (within permissions) zone values may be given as `any`, meaning there is no constraint. Note that norms only need to govern put-down actions, as each block is necessarily picked up immediately before its put-down action.

The probabilites in the grammar express prior beliefs that (a) the absence of a norm should be given a higher prior than any other potential norm (the first production rule has probability 0.5), (b) shorter obligation conditions are more probable than longer ones (note the probabilities for the rules for the *COND* non-terminal symbol), and (c) generic norms are preferred over specific ones (the `any` non-terminal symbol has a higher probability than other colour, shape and zone symbols).

5.2 Observation Likelihood

We first define the likelihood under the assumption that the observation was norm-compliant: $p(obs|\theta, \text{comp})$. Given an estimated proportion p_{nn} of non-normative executions (where an agent chooses not to be constrained by norms), the likelihood without the assumption of compliance is then $p(obs|\theta) = p_{nn}\, p(obs|\text{no-norm}) + (1 - p_{nn})\, p(obs|\theta, \text{comp})$ [Cranefield *et al.*, 2016].

[2]To allow experimentation with both simple and more complex norms, we chose to only include conditions in obligations.

The calculation of $p(obs|\theta, \text{comp})$ is necessarily domain- and grammar-specific. For our scenario, this is done as follows. When a set of observed task executions and a candidate norm expression are passed to the likelihood function, for each execution, each block b is considered in execution order. The set of compliant zones for b to be put down in is computed. In the no-norm case, there are three zones in which the block can be put down, but if the norm expression contains a prohibition or obligation, this may restrict the options. A permission can override this restriction and increase the options. In the case of an obligation for which b's colour and shape match those specified in the obligation's `next-move` condition, it is always compliant to put the block down in the zone specified by the obligation. However, placing it in the other two zones is *not* compliant if the obligation's sequence of `moved` expressions matches the task execution history. These considerations result in a set cm of compliant moves. The likelihood for the observed move of b is then 0 if its destination zone is not in cm. Otherwise the likelihood is $\frac{1}{|cm|}$. Finally the likelihoods for all the moves in the execution are multiplied to give the execution likelihood, and the execution likelihoods are multiplied to give the likelihood of the observed data. In practice, we work in log space.

6 Experiments

We first evaluated our method using the norm identification scenario and evaluation framework of Cranefield et al. [2016][3], to compare the performance of our method with their application of Bayes' Rule to maintain the odds of a finite set of norm hypotheses compared to the hypothesis that there is no norm. Their scenario involved norm-constrained travel tasks (specifying start and end nodes) on a directed graph. They considered six types of norm that constrain the sequence of nodes traversed. Instantiating these given the graph used in their experiments (actually a tree) resulted in 1932 norms, and seven of them are chosen to be the real norms known by the travelling agent but not the observer. Experimental settings include a probability of non-compliance (which we, more precisely, refer to as non-normative behaviour in the previous section), and (high) probabilities of norm-violating behaviour being observed, and being sanctioned.

Using a *behavioural* measure of precision and recall, similar to that described below (but more complex due to the presence of multiple norms), they reported the following precision P and recall R for their Bayesian norm inference method (with standard deviation over 50 runs given in brackets): $P = 64.76\,(12.24)$, $R = 95.54\,(7.16)$ for $p_{nn} = 0.01$, and $P = 67.36\,(10.12)$, $R = 85.14\,(9.54)$ for $p_{nn} = 0.3$. These were vastly better than the results for two techniques they evaluated that use data mining [Savarimuthu *et al.*, 2010; Savarimuthu *et al.*, 2013] and plan recognition [Cranefield *et al.*, 2016]. Our method achieved slight better precision and worse recall for $p_{nn} = 0.01$: $P = 67.83\,(15.42)$, $R = 89.92\,(10.34)$. For $p_{nn} = 0.3$, our approach was around 7% worse on precision and 5% worse on recall, compared to their Bayesian approach, but still far better than the other techniques.

We next applied our method to the five-block robot region-clearing task shown in Figure 2 and the norm grammar in Figure 1. The Bayesian approach described above cannot be used with a complex and recursive norm language such as this without restricting norms to a finite subset of the language. The MCMC search technique has no such constraint. We evaluated its performance in identifying an unknown "true norm" that governs an observed agent's repeated execution of the task. We chose a single true norm expression:

```
['Norms', ['Obl', ['Moved', ['Colour', 'r'],
['Shape', 'any'], ['Zone', '1'], ['Next-Move',
['Colour', 'any'], ['Shape', 'any']]], ['Zone',
'2']], ['Per', ['Action', 'putdown'], ['Colour',
'any'], ['Shape', 'square'], ['PerZone', '3']]].
```

This is the combination of (a) *an obligation to place a block in zone 2 if the move follows the placement of a red block in zone 1*; and (b) *the permission to put square blocks in zone 3* (note that the permission partially overrides the obligation).

We considered values of p_{nn} between 0 and 0.55, in increments of 0.05. For each p_{nn}, we ran three trials of the following experiment. We generated a sequence of "observed" random task executions that were intentionally compliant with the norm with probability $1 - p_{nn}$. These executions varied in terms of the order of blocks moved and the zones to which the blocks were moved. Our Metropolis-Hastings algorithm (with the modified acceptance from equation 1, and $\alpha = 0.1$) was run to generate ten chains of length 4800. After discarding the first half of each chain, and splitting the remaining chain into two, the chain convergence metric $\hat{R}$ was iteratively calculated over subsequences of the resulting 20 chain segments that doubled in length until the end of the segments. The segments were then combined to form a sample of the posterior distribution over expressions. Observation likelihoods, for each of the expressions featuring in the chain, were calculated as part of the algorithm, and these were combined with the expression's prior probabilities (from the grammar) to calculate the (log) posterior probabilities. These were stored along with the expressions in the posterior sample. Thus, our approach allows selecting candidate norms either through their frequency in the posterior sample or by using their posterior probabilities.

Figure 4 shows the log posterior of expressions in the posterior sample, and their rank by frequency, for all trials, plotted against p_{nn}. The subplots use shapes to distinguish different trials. In the top plot, pink shapes show the maximum log posterior within the posterior sample for that trial, and green ones show the log posterior of the true norm. Green above pink for the same shape indicates that the true norm was not within the posterior sample. Pink above green occurs in some cases where $p_{nn} \geq 0.25$. This indicates that an expression in the sample has a posterior higher than the true norm, and therefore is a better explanation of the observed behaviour due to the high rates of non-normative behaviour. In all such cases, this expression was the "No-norm" expression.

[3]Their code is available at https://github.com/mir-pucrs/norm-detect

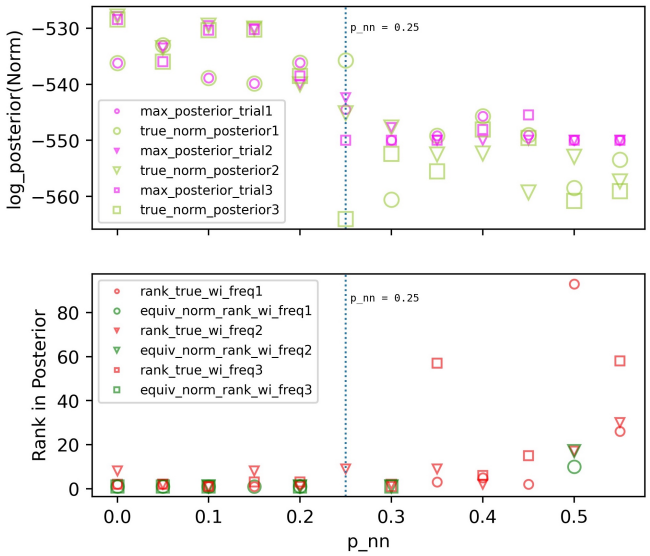

Figure 4: Expression log posteriors and ranks by frequency within the posterior sample

p_{nn}	**precision**	**recall**	p_{nn}	**precision**	**recall**
0.00	0.920	0.974	0.30	0.898	0.974
0.05	0.896	0.974	0.35	0.773	0.878
0.10	0.977	0.977	0.40	0.894	0.909
0.15	0.919	0.941	0.45	0.815	0.938
0.20	0.976	0.977	0.50	0.881	0.924
0.25	0.839	0.939	0.55	0.816	0.938

Table 1: Precision and recall for the most frequent norm in the posterior, averaged over three trials

The lower plot shows the rank in the posterior sample of the true norm and the highest ranked equivalent norm (if present). For higher values of p_{nn}, some trials found neither the true norm nor an equivalent one. The variation in rank across trials increases with p_{nn}, showing that selecting a norm by log posterior rather than by rank in the sample, is a better approach in these cases.

We now consider the precision and recall of the identified norm, when selected by frequency rank. We do not use a standard definition of these concepts based on a simple identity test between true and identified expressions. This is because, given a particular task, two or more norm expressions may be equivalent in terms of the constraints they apply to executions of the task. For example, consider a prohibition against putting down green squares in zone 1. For the task in Figure 2, the logically more general prohibition against putting down *any* green block in zone 1 is, in fact, equivalent to the former expression, because the *only* green block to be moved is square. Therefore, we use *behavioural* interpretations of precision and recall [Cranefield *et al.*, 2016]. These measure how well an agent governed by the candidate norm can generate task executions that are compliant with the true norm expression. We generated two sets of 100,000 random norm-compliant task executions: for the true norm and the candidate norm expression. We treated the former set as it were the complete set of true-norm-compliant executions when counting 'true' and 'false' positives in the latter set. Note that as we did not generate all possible true-norm-compliant task executions, the results give approximations to the behavioural precision and recall.

Table 1 shows precision and recall for different values of p_{nn}, assuming that the most frequent norm in the posterior sample is used to govern the observer's own behaviour. Even though the most frequent norm may not be the true norm, or even an equivalent, high precision and recall are seen in almost all cases.

Finally, we consider the convergence of the chains in each experiment. We find that, on average across the trials and values of p_{nn}, the $\hat{R}$ statistic increased from 1.05 to 1.24 (rather than decreasing towards 1) as the initial segments of the chains evaluated increased in length from 50 to 1200. There were no notable differences across the cases. This indicates a lack of convergence, and points to a possible need for longer chains. However, if the aim of the observer is solely to adopt the expression with the highest log posterior probability as the norm, rather than create an accurate posterior sample to use for posterior predictive inference, then the use of multiple MCMC chains is a successful search mechanism.

7 Conclusion

This paper presents an adaptation of the prior work of Saad et al. [2019] to the problem of norm identification from observations. We have presented their method as an application of the Metropolis-Hastings algorithm, and modified it to favour norms that are relevant to the task being observed. We also highlighted the importance of analysing the convergence of the MCMC chains generated (not considered, to our knowledge, in prior research on MCMC search over symbolic expressions). We propose the use of a vector representation of trees, based on the tree kernel of Collins and Duffy [2002], to define a metric over expressions and allow the $\hat{R}$ convergence statistic to be applied in our work. Unlike previous work on Bayesian methods in norm identification, our approach does not require selecting a pre-determined finite set of candidate norms. The recursive norm language may be defined to allow arbitrarily complex norms, as illustrated by the obligation norms used in this work.

Our experiments showed that for our experimental scenario, when the probability of non-normative behaviour does not exceed 0.25, MCMC search allowed the true norm expression, or an equivalent one, to be identified as the expression with the highest log posterior probability. Norm identification based on the most frequent expression in the posterior sample can also be a successful means of generating compliant behaviour, based on our precision and recall results, even if the identified norm is not the true one.

Futher research is needed on improving the efficiency of MCMC search for norms, and determining an $\hat{R}$ threshold for "good enough" convergence. It would also be useful to investigate other forms of probabilistic grammar that are more apt for encoding norms, and to adapt the approach used in this paper for incremental learning.

References

[Andrighetto *et al.*, 2013] Giulia Andrighetto, Guido Governatori, Pablo Noriega, and Leendert W. N. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

[Campos *et al.*, 2010] Jordi Campos, Maite López-Sánchez, and Marc Esteva. A case-based reasoning approach for norm adaptation. In *International Conf. on Hybrid Artificial Intelligence Systems*, pages 168–176. Springer, 2010.

[Chaudhuri *et al.*, 2019] Ritwik Chaudhuri, Kushal Mukherjee, Ramasuri Narayanam, Rohith Dwarakanath Vallam, Ayush Kumar, Antriksh Mathur, Shweta Garg, Sudhanshu Singh, and Gyana R. Parija. Collaborative reinforcement learning model for sustainability of cooperation in sequential social dilemmas. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 1877–1879. IFAAMAS, 2019.

[Collins and Duffy, 2002] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2002.

[Cranefield *et al.*, 2016] Stephen Cranefield, Felipe Meneguzzi, Nir Oren, and Bastin Tony Roy Savarimuthu. A Bayesian approach to norm identification. In *22nd European Conference on Artificial Intelligence*, pages 622–629. IOS Press, 2016.

[Dellarocas and Klein, 2000] Chrysanthos Dellarocas and Mark Klein. Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces. In *Agent Mediated Electronic Commerce II*, pages 24–39. Springer, 2000.

[Gelman *et al.*, 2013] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, third edition, 2013.

[Gulwani *et al.*, 2017] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends in Programming Languages*, 4(1-2):1–119, 2017.

[Lee and Wagenmakers, 2014] Michael D. Lee and Eric-Jan Wagenmakers. *Bayesian Cognitive Modeling: A Practical Course*. Cambridge University Press, 2014.

[Morales *et al.*, 2013] Javier Morales, Maite López-Sánchez, Juan A. Rodríguez-Aguilar, Michael J. Wooldridge, and Wamberto Weber Vasconcelos. Automated synthesis of normative systems. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 483–490. IFAAMAS, 2013.

[Morales *et al.*, 2015] Javier Morales, Maite López-Sánchez, Juan Antonio Rodríguez-Aguilar, Wamberto Weber Vasconcelos, and Michael J. Wooldridge. Online automated synthesis of compact normative systems. *ACM Transactions on Autonomous and Adaptive Systems*, 10(1):2:1–2:33, 2015.

[Oren and Meneguzzi, 2013] Nir Oren and Felipe Meneguzzi. Norm identification through plan recognition. 15th International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems, arXiv:2010.02627, 2013.

[Royakkers, 1997] Lamber M. M. Royakkers. Giving permission implies giving choice. In *Proceedings of the 8th International Conference on Database and Expert Systems Applications*, pages 198–203. IEEE, 1997.

[Saad *et al.*, 2019] Feras A. Saad, Marco F. Cusumano-Towner, Ulrich Schaechtle, Martin C. Rinard, and Vikash K. Mansinghka. Bayesian synthesis of probabilistic programs for automatic data modeling. *Proceedings of the ACM on Programming Languages*, 3: 37:1–37:32, 2019.

[Sarathy *et al.*, 2017] Vasanth Sarathy, Matthias Scheutz, and Bertram F. Malle. Learning behavioral norms in uncertain and changing contexts. In *Proceedings of the 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 301–306, 2017.

[Savarimuthu and Cranefield, 2011] Bastin Tony Roy Savarimuthu and Stephen Cranefield. Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, 7(1):21–54, 2011.

[Savarimuthu *et al.*, 2010] Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam Purvis, and Martin K. Purvis. Obligation norm identification in agent societies. *Journal of Artificial Societies and Social Simulation*, 13(4), 2010.

[Savarimuthu *et al.*, 2013] Bastin Tony Roy Savarimuthu, Stephen Cranefield, Maryam Purvis, and Martin K. Purvis. Identifying prohibition norms in agent societies. *Artificial Intelligence and Law*, 21(1):1–46, 2013.

[Sen and Airiau, 2007] Sandip Sen and Stéphane Airiau. Emergence of norms through social learning. In *20th International Joint Conference on Artificial Intelligence*, pages 1507–1512, 2007.

[Tan *et al.*, 2019] Zhi-Xuan Tan, Jake Brawer, and Brian Scassellati. That's mine! learning ownership relations and norms for robots. In *Proceedings of the The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 8058–8065. AAAI Press, 2019.

[Wang *et al.*, 2019] Jane X. Wang, Edward Hughes, Chrisantha Fernando, Wojciech M. Czarnecki, Edgar A. Duéñez Guzmán, and Joel Z. Leibo. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 683–692. IFAAMAS, 2019.

[Zhang *et al.*, 2019] Chengwei Zhang, Xiaohong Li, Jianye Hao, Siqi Chen, Karl Tuyls, Wanli Xue, and Zhiyong Feng. SA-IGA: a multiagent reinforcement learning method towards socially optimal outcomes. *Autonomous Agents and Multi-Agent Systems*, 33:403–429, 2019.