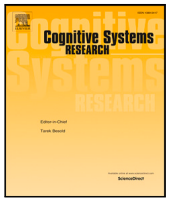
Cognitive Systems Research 86 (2024) 101232

Contents lists available at ScienceDirect

Cognitive Systems Research

journal homepage: www.elsevier.com/locate/cogsys



How do you know that you don't know?☆

Quentin F. Gronau [a,*], Mark Steyvers [b], Scott D. Brown [a]

[a] School of Psychological Sciences, The University of Newcastle, Australia
[b] Department of Cognitive Sciences, University of California, Irvine, United States of America

ARTICLE INFO

Action editor: Serge Thill

Dataset link: https://osf.io/t4n63/

Keywords:
Model uncertainty
Model mis-specification
Maximum a posteriori probability

ABSTRACT

Whenever someone in a team tries to help others, it is crucial that they have some understanding of other team members' goals. In modern teams, this applies equally to human and artificial ("bot") assistants. Understanding when one does not know something is crucial for stopping the execution of inappropriate behavior and, ideally, attempting to learn more appropriate actions. From a statistical point of view, this can be translated to assessing whether none of the hypotheses in a considered set is correct. Here we investigate a novel approach for making this assessment based on monitoring the maximum a posteriori probability (MAP) of a set of candidate hypotheses as new observations arrive. Simulation studies suggest that this is a promising approach, however, we also caution that there may be cases where this is more challenging. The problem we study and the solution we propose are general, with applications well beyond human–bot teaming, including for example the scientific process of theory development.

1. Introduction

Being aware of another agent's goals is a key developmental stage for children as they grow up and develop a *theory of mind*. This awareness is crucial for being able to successfully interact with others and collaborate, not just as children, but at any age. For instance, being able to help another requires the helping person to know about the goal of the other person and the possible actions for achieving that goal (Kumar & Steyvers, 2022; Puig et al., 2020). Suppose you see a person leaving money on the table in a restaurant. Should you try to return the money, because you believe the person left it unintentionally, or should you leave it, because the person intentionally left the money as a tip? If you misunderstand the other person's goal in this scenario, you might erroneously try to return the money.

Awareness of another agent's goals is not only important in human interactions, but also in human–bot settings. For instance, autonomous driving requires the bot to accurately predict the trajectories of pedestrians, cyclists, and other cars, to avoid collisions (Fridovich-Keil et al., 2020). However, what if a pedestrian moves in an unexpected way? For instance, what if a pedestrian tries to avoid a spill on the ground that is unaccounted for by the bot, leading to inaccurate predictions by the bot and, possibly, collision? In this scenario, it is key for the bot to realise in time that it does not understand the goal of the pedestrian so that it can be more cautious or even stop driving.

As another example, suppose a bot tries to assist a human in sorting objects into target and non-target categories (Fig. 1). The bot is repeatedly presented with objects that vary in three features: color, shape, and the number written on them. The bot's task is to infer the rule the human uses for selecting the objects so that it can successfully help sorting the objects. The rule could be simple, such as "red objects", but could also be more complex, such as "red triangular objects", or "objects which are not blue and are not square", or even something complex such as "make sure there are more triangles in the target category than the non-target category". Even with this simplified set-up, the space of possible objectives that are consistent with the data is very large. Suppose each feature type has three possible levels: "1", "2", and "3". For instance, for the feature type "color", "1" might correspond to red, "2" to green, and "3" to blue. Suppose the bot only has a sensor to detect the color of the presented objects, but it cannot detect the shape or the number written on the objects. In this case, there are only three possible hypotheses available to the bot: (1) red objects, (2) green objects, (3) blue objects. But what if the true rule is more complicated? For instance, even if the rule is just "red triangular objects", none of the bot's candidate hypotheses are correct, resulting in the bot hindering rather than helping to sort the objects.

These examples highlight that it is important to find ways of knowing when one does *not* know. The problem can be framed very generally

☆ This work was partially supported by the Australian Research Council (DP210100313) and the Australian Defence Science and Technology Group (AUSMURIIV000001).

* Correspondence to: School of Psychological Sciences, 2308 Callaghan Campus, New South Wales, Australia.
E-mail address: Quentin.F.Gronau@gmail.com (Q.F. Gronau).

https://doi.org/10.1016/j.cogsys.2024.101232
Received 23 May 2023; Received in revised form 11 March 2024; Accepted 14 March 2024
Available online 19 March 2024

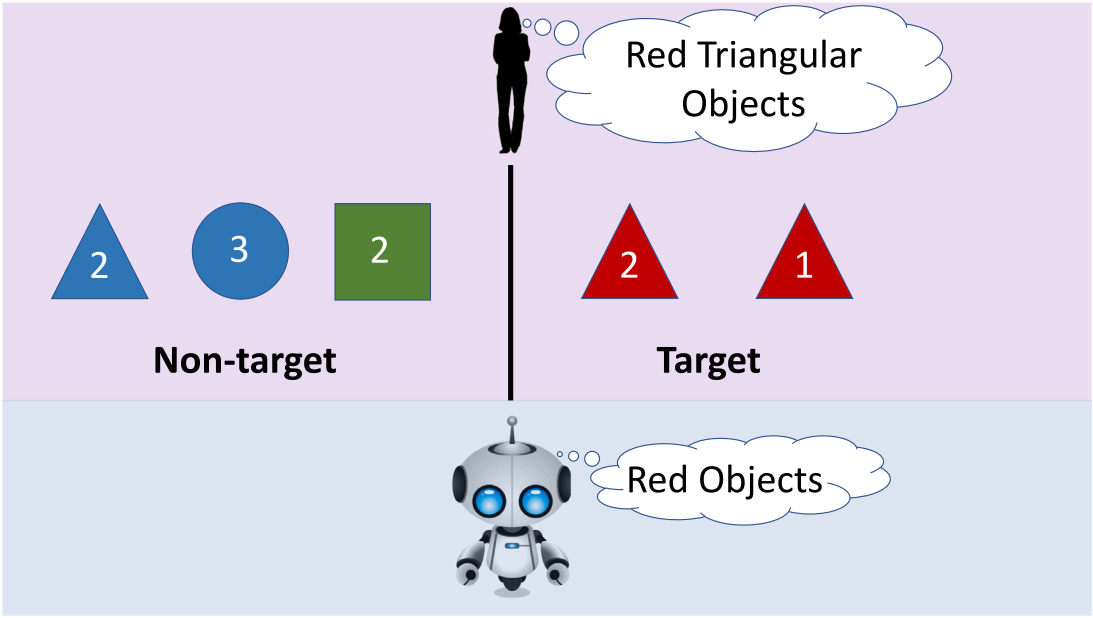


Fig. 1. A bot tries to assist a human in sorting objects into target and non-target categories. The bot is repeatedly presented with objects that vary in three features: color, shape, and the number written on them. The bot's task is to infer the rule the human uses for selecting the objects so that it can successfully help sorting the objects. In this example, the human uses the rule "red triangular objects". However, the bot can only detect the color of the presented objects, but not the shape or number written on the objects. Consequently, there are only three possible hypotheses available to the bot: (1) red objects, (2) green objects, (3) blue objects. Out of these available hypotheses, the bot believes it is most likely that the rule is "red objects". However, the true rule used by the human is more complicated (i.e., "red triangular objects").

by building on the hypothesis testing language above. One agent (the bot, in our case) considers a set of hypotheses about the goal of the other agent (the human being assisted). Standard statistical approaches allow the posterior probabilities of hypotheses in the set to be updated as new information arrives, eventually resulting in one of the hypotheses being chosen. Our interest is in how the bot could come to realize that none of the hypotheses in the consideration appropriately represent the human's goal. Similar to theories of metamemory that assume a separate metacognitive level that monitors lower-level memory activity (Nelson, 1990), we propose a second, metacognitive, decision process (Fig. 2). The lower-level decision process which evaluates the considered hypotheses against incoming data is specific to the problem at hand, but we are interested in exploring more general metacognitive processes, which might apply to a wide range of situations. The role of the metacognitive decision process is to infer, from observing only the statistical properties of the considered hypotheses, whether none of those hypotheses is correct.

Detecting a mismatch between data and *all* considered hypotheses is relevant not only in the scenarios described above, but also in the ubiquitous case of adjudicating between competing scientific theories by means of quantitative model comparison metrics. In that case, ideally one would like to assess not only which theory is best supported by the data, but also if none of the theories – not even the best in the considered set – is appropriate. Unfortunately, knowing that none of the models or hypotheses in a considered set is correct is far from trivial. The processes of expanding one's hypothesis space in the face of mis-specification has been studied in an effort to characterise creativity (e.g. Boden, 2009; McGreggor, Kunda, & Goel, 2011), but the moment that triggers this creative expansion has not.

2. Previous work

Coming to the realisation that one needs to expand their hypothesis space in face of mis-specification bears resemblance to insight problems that have been studied in psychological research (Chu & MacGregor, 2011). For instance, the task could be to use exactly six pencils (without breaking or bending any) to construct four triangles. Faced with such a problem, the problem solver typically starts with an incorrect representation which does not allow for a solution. For instance, they may search for a solution in the 2D plane. Similar to realising that no hypothesis in the considered set is correct, the problem solver must come to the insight that their initial representation is incorrect and change their representation. In this particular example, the problem solver must realise the solution may exist in 3D space to come up with the correct solution, a tetrahedron. In contrast to insight problems where the insight is a creative process that is difficult to plan ahead, to be of practical use in a human–bot interaction where the bot needs to decide whether none of its known hypotheses are correct, a more systematic approach is needed.

Another link that can be drawn is with research on recognition memory. Specifically, in a typical recognition memory experiment, participants study a list of words in a study phase. Subsequently, in the test phase, participants are presented with one word at a time and they have to decide whether the presented word was part of the study list ("old" word) or not ("new" word). This set-up has parallels with the bot's task to decide whether none of its considered hypotheses are correct: As opposed to recognising whether a presented word has been part of the study list or not, the bot's task is to recognise whether the observed data have originated from a known or unknown hypothesis. In fact, as elaborated on below, the key quantity of our proposed algorithm draws upon components of one of the most successful models for recognition memory (Shiffrin & Steyvers, 1997).

Related problems have also been studied in the context of machine learning rather than cognitive science, as *out-of-distribution* (OOD) detection (Yang, Zhou, Li, & Liu, 2021). The goal of OOD detection is to equip a bot with methods for detecting unknown observations. For instance, a bot trained to classify food should be able to detect and reject non-food images such as selfies uploaded by users instead of blindly classifying them into existing food categories. However, our goal here is more ambitious than detecting single unknown observations. Instead, we investigate how a bot could detect that the overarching rule for how observations come about, the data-generating hypothesis, is not part of its considered set. To highlight this difference, consider the example presented above where a bot assists a human in sorting objects

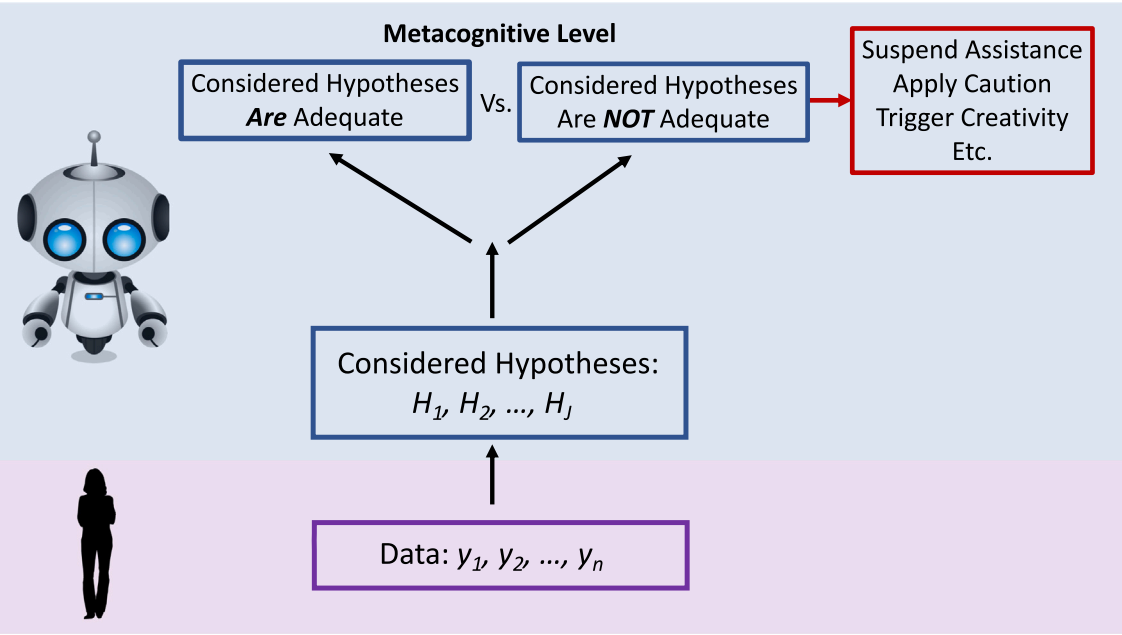


Fig. 2. We investigate a metacognitive process which monitors inferential decision making to detect when none of the hypotheses being considered are appropriate.

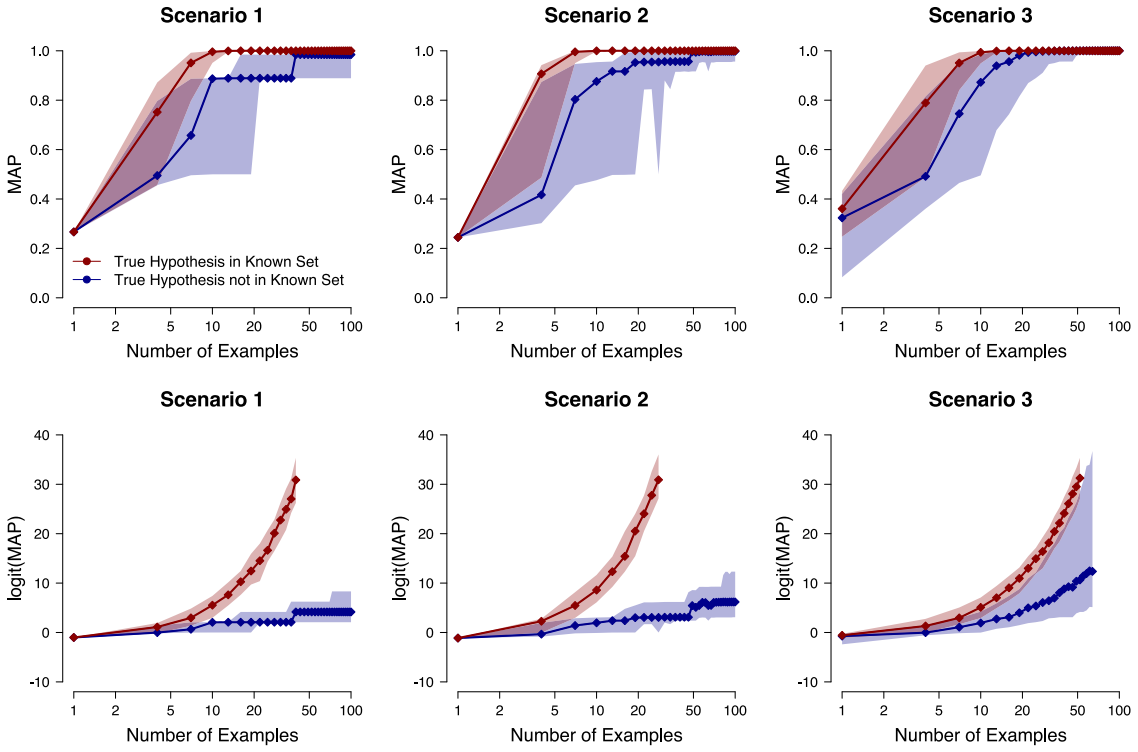


Fig. 3. The upper row displays, for each scenario separately, the median of the MAP (dots), accompanied by the 25% and 75% quantiles (shaded area), as a function of the number of presented examples (on a log scale), split by whether the true hypothesis was in the considered set (red symbols) or not (blue symbols). The bottom row displays the same data but with a logit-transformed y-axis.

into target and non-target categories. The considered bot only has a color sensor and believes that the three possible colors that a presented object could have are red, green, and blue. An example of an out-of-distribution observation would be an object that has none of these three colors, but is, for instance, yellow. However, in many scenarios there may not exist single observations that are such clear indicators of the fact that the human uses a rule unknown to the bot. For instance, when the human uses the rule "red triangular objects", none of the presented

objects is a clear indication of the fact that the bot is wrong, but it is still crucial for the bot to realise that it does not understand the data-generating rule.

Decisions about one's own ignorance have also been studied in the context of human memory (Glucksberg & McCloskey, 1981), for instance, deciding with high confidence that one does not know the answer to a general knowledge question. However, to the best of our knowledge, in the context of human–bot interactions, the question of how a bot could know that they don't know has so far not been investigated in great depth. Nevertheless, recently there has been some promising work on this subject.

Fridovich-Keil et al. (2020) considered the case of robot motion planning and proposed that rather than trusting a model's predictions blindly, the robot should use the model's current predictive accuracy to inform the degree of confidence in its future predictions. This allows for probabilistic motion predictions that exploit modeled structure when the structure successfully explains human motion, and degrade gracefully whenever the human moves unexpectedly. Their implementation focuses on Bayesian belief over a single parameter that governs the variance of a human motion model.

Bobu, Bajcsy, Fisac, Deglurkar, and Dragan (2020) explicitly considered the case when a robot's hypothesis space is mis-specified. They proposed that the robot should reason about how well it can explain human inputs given its hypothesis space and then use that situational confidence to inform how it should incorporate human input. In their implementation, this was achieved by the robot making inferences about the human's cost function.

In a purely statistical context, one approach to knowing whether one does not know is to consider tests of absolute fit or model mis-specification. Given a set of candidate hypotheses or models, one could then test whether the best out of these is still mis-specified, suggesting that none of the considered hypotheses or models is correct. A range of tests for model mis-specification have been proposed in the statistical literature, many based on the information matrix equality (e.g., Golden, Henley, White, & Kashner, 2016), but also based on cross-entropy (Krauz & Tabrikian, 2019), and in-sample and out-of-sample likelihoods (Presnell & Boos, 2004).

3. Proposed approach

Here we propose a novel approach for how the bot could come to realize that none of the hypotheses in the consideration appropriately represent the human's goal. Our approach focuses on the maximum posterior probability over hypotheses in the considered set as a function of the number of observations.

Suppose a bot or human considers J hypotheses: $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_J$. Based on observed data $\boldsymbol{y}$ featuring n observations, they can then update their prior beliefs about how likely each of these hypotheses is to posterior beliefs using Bayes' rule as follows:

$$p(\mathcal{H}_j \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \mathcal{H}_j)\, p(\mathcal{H}_j)}{\sum_{k=1}^{J} p(\boldsymbol{y} \mid \mathcal{H}_k)\, p(\mathcal{H}_k)}, \tag{1}$$

where $p(\mathcal{H}_j)$ denotes the prior probability of hypothesis $\mathcal{H}_j$, $p(\mathcal{H}_j \mid \boldsymbol{y})$ the corresponding posterior probability after having observed data $\boldsymbol{y}$, and $p(\boldsymbol{y} \mid \mathcal{H}_j)$ is the probability of the data given the hypothesis (also referred to as *marginal likelihood*). In many applications, the prior probability of each hypothesis $\mathcal{H}_j$ is set equal (i.e., $p(\mathcal{H}_j) = 1/J$), reflecting an a priori position of indifference. This is often referred to as a state of maximum ignorance.

It is well-known that if the true data-generating hypothesis is in the considered set, the posterior probability for this hypothesis will go to 1 as the number of observations increases (Chatterjee, Maitra, & Bhattacharya, 2020, Corollary 2). If the true hypothesis is not in the considered set, eventually, the posterior probability for the hypothesis that is closest to the true one (in a KL-divergence sense) will go to 1 (Chatterjee et al., 2020, Corollary 2). However, as demonstrated below, this convergence may often happen more slowly than when the true hypothesis is in the considered set. Consequently, we propose that one can obtain an assessment of whether the true hypothesis is in the considered set by investigating how quickly the maximum a posteriori probability (MAP) approaches 1 as the number of observations n increases.

Table 1
Hypotheses.

Hypothesis no.	Feature 1	Feature 1 value	Feature 2	Feature 2 value
1	1	1	–	–
2	1	2	–	–
3	1	3	–	–
4	2	1	–	–
5	2	2	–	–
6	2	3	–	–
7	3	1	–	–
8	3	2	–	–
9	3	3	–	–
10	1	1	2	1
11	1	1	2	2
12	1	1	2	3
13	1	2	2	1
14	1	2	2	2
15	1	2	2	3
16	1	3	2	1
17	1	3	2	2
18	1	3	2	3
19	1	1	3	1
20	1	1	3	2
21	1	1	3	3
22	1	2	3	1
23	1	2	3	2
24	1	2	3	3
25	1	3	3	1
26	1	3	3	2
27	1	3	3	3
28	2	1	3	1
29	2	1	3	2
30	2	1	3	3
31	2	2	3	1
32	2	2	3	2
33	2	2	3	3
34	2	3	3	1
35	2	3	3	2
36	2	3	3	3

4. Inference about an unknown rule

Next we present results of a simulation study in which we assume that a bot assists a human in sorting objects into targets and non-targets, adopting a setup similar to the one described in Fig. 1.

4.1. Hypotheses

For concreteness, we consider the case where the bot entertains two types of hypotheses: (1) one-feature hypotheses (specific color, specific shape, or specific number) and (2) two-feature hypotheses (e.g., specific color + specific shape, specific shape + specific number). An example of a one-feature hypothesis would be "red objects" that could have any shape or number. An example of a two-feature hypothesis would be "red triangular objects" with any number. We can represent the 36 possible hypotheses as shown in Table 1. There are nine one-feature hypotheses (numbered 1–9 in the table) and 27 two-feature hypotheses (10–36).

4.2. Data-generating process

The bot is presented with n objects selected based on a rule unknown to it. There is some noise in the process, such that it is possible

that some objects will not perfectly correspond to the underlying rule. Each of the n objects that are presented can be represented by a vector $\boldsymbol{y}_i = (y_{1,i}, y_{2,i}, y_{3,i})$, where $y_{1,i}$ denotes the value for feature 1, $y_{2,i}$ the value for feature 2, and $y_{3,i}$ denotes the value for feature 3.

4.2.1. One-feature hypotheses

If the true data-generating rule is a one-feature hypothesis, with probability p_c, the observed value for the feature of interest is set to the correct value.[1] For instance, for hypothesis no. 3 which states that feature 1 has value 3, $y_{1,i} = 3$ with probability p_c. With probability $1-p_c$ the value for the feature of interest is chosen randomly (i.e., probability 1/3 for each of the three possible values). Consequently, the total probability that a feature has its correct value is given by $p_c + (1 - p_c)\frac{1}{3}$. The values for the other features that are not specified by the data-generating hypothesis are chosen randomly (i.e., probability 1/3).

4.2.2. Two-feature hypotheses

The two features of interest are jointly set to their correct value with probability p_c. Otherwise the data-generating mechanism is a straightforward extension of the one for one-feature hypotheses.[2]

4.3. Scenarios

Next we investigate a number of scenarios where the bot considers only a subset of all possible true data-generating hypotheses. Consequently, sometimes it is possible for the bot to infer the correct data-generating hypothesis since it is part of the considered set, but sometimes this is impossible since the true hypothesis is not part of the set. We are interested in investigating whether it is possible for the bot to know if the latter is the case based on considering the MAP as described above.

4.3.1. Scenario 1: One-feature hypotheses

In the first scenario, we assume the bot considers only one-feature hypotheses. However, the true data-generating hypothesis could be any hypothesis from Table 1. That is, the true data-generating hypothesis may also be a two-feature hypothesis.

4.3.2. Scenario 2: Two-feature hypotheses

In the second scenario, we assume the bot considers only two-feature hypotheses. However, the true data-generating hypothesis could be any hypothesis from Table 1. That is, the true data-generating hypothesis may also be a one-feature hypothesis.

4.3.3. Scenario 3: Mixed hypotheses $\mathcal{P}_1$

In the first two scenarios, the *considered* hypotheses are all of equal complexity: either all of them are one-feature hypotheses (Scenario 1) or two-feature hypotheses (Scenario 2). In contrast, the *true* data-generating hypotheses are of mixed complexity (i.e., both one-feature and two-feature hypotheses). In Scenario 3 we investigate whether considering the convergence rate of the MAP is a useful indicator for whether the true hypothesis is in the considered set for the case where the considered set contains both one-feature and two-feature hypotheses. We split the 36 hypotheses from Table 1 into three parts, each containing three one-feature and nine two-feature hypotheses. The first, part which we refer to as $\mathcal{P}_1$, contains hypotheses no. 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34. The second part, $\mathcal{P}_2$, contains hypotheses no. 2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35. Finally, the third part, $\mathcal{P}_3$, contains hypotheses no. 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36. Here we do not consider $\mathcal{P}_3$. Instead, we focus on the case were the true data-generating hypothesis could be any hypothesis that is part of either $\mathcal{P}_1$ or $\mathcal{P}_2$. In this scenario, we assume that the bot considers only $\mathcal{P}_1$ hypotheses.

[1] In this simulation study, we set $p_c = 0.7$.

[2] Note that the computation of the likelihoods assumes a strong sampling process where objects are sampled randomly from all objects consistent with the data generating rule (Tenenbaum & Griffiths, 2001).

4.4. Simulation details

For each of the three scenarios, we simulated data from each of the possible generating hypotheses and monitored the MAP for the considered set of hypotheses as a function of the number of presented examples. We repeated this process 100 times for each scenario.

4.5. Results

Fig. 3 displays the results of the simulation study. Each panel in the upper row shows the median of the MAP (dots), accompanied by the 25% and 75% quantiles (shaded area), as a function of the number of presented examples (on a log scale), split by whether the true hypothesis was in the considered set (red symbols) or not (blue symbols). The bottom row displays the logit transformed MAP as a function of the number of presented examples in a similar manner (i.e., median accompanied by the 25% and 75% quantiles).

For all three scenarios, it is apparent that when the true data-generating hypothesis is in the considered set (red), the MAP (Fig. 3, upper row) converges to 1 more quickly than when the true data-generating hypothesis is not in the considered set (blue). This finding is most pronounced for Scenario 1 and Scenario 2. For Scenario 3, the MAP trajectory also differs when the true hypothesis is in the considered set compared to when it is not, however, the difference is smaller than in the other two scenarios. This suggests that there may also exist cases where discerning whether or not the true hypothesis is in the considered set based on the MAP trajectory may be more challenging.

The trajectory of the logit transformed MAP (Fig. 3, bottom row) highlights that, for all three scenarios, convergence is faster when the true hypothesis is in the considered set. Importantly, plotting the logit transformed MAP highlights that this difference in fact increases as the number of presented examples increases. In the Appendix, we report results for a similar simulation that only differed in that the probability of setting the feature(s) of interest to their correct values, p_c, was a free parameter and not fixed. The results are very similar to the ones for the case reported here where p_c was fixed to 0.7.

5. Applying the approach in practice

After having demonstrated that the MAP trajectory can in certain scenarios be a useful tool for assessing whether the true hypothesis is in the set, we next propose an algorithm for how a bot could use this finding in practice to inform decisions about whether they don't know and should therefore stop assisting, ask for help, etc.

5.1. Algorithm

The proposed algorithm (Algorithm 1) assumes that the bot considers a set of candidate hypotheses $\boldsymbol{\mathcal{H}_c} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_J\}$ and is able to simulate data from each hypothesis in $\boldsymbol{\mathcal{H}_c}$. In a *training phase*, the bot repeatedly simulates the scenario that there is one unknown hypothesis by repeatedly randomly choosing one hypothesis to be the "unknown" one in a leave-one-out (LOO) manner. The remaining hypotheses are considered to be "known" and serve as the candidate set of hypotheses. The bot then simulates data from *all* hypotheses (i.e., also the one that has been chosen to be "unknown") and computes the MAP trajectory based on posterior probabilities for the $J - 1$ hypotheses that are considered to be "known". The bot saves the MAP trajectory either in a "known" category, in case the data-generating hypothesis was in the "known" set, or in an "unknown" category, in case the data-generating hypothesis was the "unknown" one. After having repeated this process a number of times, the bot then fits a location-scale regression model separately to the MAP trajectories saved in the "known" and "unknown" categories. Specifically, the dependent variable corresponds to the appropriately transformed MAP values and the independent

Algorithm 1

1. **Training phase:**
   (a) For $i = 1$ to n_{reps}:
      i. Randomly select one hypothesis from the candidate set $\mathcal{H}_c$ and assign it the label "unknown". The set of the remaining $J-1$ hypotheses is denoted by $\mathcal{H}_{c^*}^{(i)}$.
      ii. For $j = 1$ to n_{sims}:
         A. Generate a data sequence of length n_{train} from each of the hypotheses in $\mathcal{H}_c$.
         B. For each of these, compute the MAP trajectory based on posterior probabilities computed only for the hypotheses in $\mathcal{H}_{c^*}^{(i)}$.
         C. Save the MAP trajectory for each of these with the label "known" if the generating hypothesis was in $\mathcal{H}_{c^*}^{(i)}$ and "unknown" otherwise.
   (b) Fit a location-scale regression model separately to the MAP trajectories with "known" and "unknown" label. The predictor is the log of the number of observations, the dependent variable the appropriately transformed MAP trajectories.
2. **Application phase:**
   (a) Observe n data points.
   (b) Compute the MAP trajectory based on posterior probabilities computed for the hypotheses in $\mathcal{H}_c$.
   (c) Compute LR_{ku} as described in Equation (2).
   (d) If $\text{LR}_{ku} < C$, conclude don't know.

variable is the log of the number of observations. One option for the transformation function f is the logit function which was used in the previous section. However, for the following simulation, we used $f(x) = -\log(-\log(x))$ which improved the numerical stability of the algorithm.

Using these two regression fits, the bot is now in the position to observe a MAP trajectory based on real data and conduct a likelihood ratio test to assess whether this MAP trajectory corresponds to a known hypothesis or an unknown hypothesis that is not part of the set. Specifically, in the *application phase*, the bot observes n real data points with corresponding MAP values $\text{MAP}_1, \text{MAP}_2, \ldots, \text{MAP}_n$. The likelihood ratio of interest is then given by:

$$\begin{aligned}\text{LR}_{ku} &= \prod_{i=1}^{n} \frac{p(\text{MAP}_i \mid \text{known})}{p(\text{MAP}_i \mid \text{unknown})} \\ &= \prod_{i=1}^{n} \frac{\mathcal{N}(f(\text{MAP}_i); \mu_i^{\text{known}}, \sigma_i^{\text{known}})}{\mathcal{N}(f(\text{MAP}_i); \mu_i^{\text{unknown}}, \sigma_i^{\text{unknown}})},\end{aligned} \tag{2}$$

where μ_i^{known}, σ_i^{known} correspond to the predicted values for the location and scale, respectively, for the "known" location-scale regression fit and μ_i^{unknown}, $\sigma_i^{\text{unknown}}$ to the ones for the "unknown" fit. Here, $\mathcal{N}(x; \mu, \sigma)$ denotes the density of a normal distribution with mean μ and standard deviation σ evaluated at x. Values of $\text{LR}_{ku} > 1$ indicate evidence in favor of the MAP trajectory originating from a known hypothesis that is part of the considered set. In contrast, values of $\text{LR}_{ku} < 1$ indicate evidence in favor of the MAP trajectory originating from an unknown hypothesis. The smaller LR_{ku}, the more evidence the data provide in favor of the data originating from an unknown hypothesis. The bot may choose to set a criterion value C such that it concludes that it does not know the data-generating hypothesis and stops assisting the human if $\text{LR}_{ku} < C$. The exact setting of C may depend on risk-benefit trade-offs. For instance, if the bot is a self-driving car, it may err on the side of being overly cautious (i.e., C may be close to or exactly equal to 1) whereas in other scenarios it may be more costly for the bot to stop assisting the human too early (i.e., C may be set lower such that assistance is stopped only when the evidence is overwhelming that the data originate from an unknown hypothesis).

An attractive property of using the likelihood ratio LR_{ku} is that it allows the bot to monitor the evidence in favor of the data originating from a known vs. unknown hypothesis in real time. Specifically, the likelihood ratio in Eq. (2) is itself given by the product of n likelihood ratios. Each of these individual likelihood ratios assesses, for a single MAP value, how likely it is that it is based on a scenario where the true hypothesis is known vs. unknown. Consequently, it is straightforward and computationally efficient to update LR_{ku} when a new observation arrives – hence, also a new MAP value, MAP_{n+1} – by simply multiplying the current value of LR_{ku} by the individual likelihood ratio for the new observation:

$$\begin{aligned}\text{LR}_{ku}^{(n+1)} &= \text{LR}_{ku}^{(n)} \times \frac{p(\text{MAP}_{n+1} \mid \text{known})}{p(\text{MAP}_{n+1} \mid \text{unknown})} \\ &= \text{LR}_{ku}^{(n)} \times \frac{\mathcal{N}(f(\text{MAP}_{n+1}); \mu_{n+1}^{\text{known}}, \sigma_{n+1}^{\text{known}})}{\mathcal{N}(f(\text{MAP}_{n+1}); \mu_{n+1}^{\text{unknown}}, \sigma_{n+1}^{\text{unknown}})},\end{aligned} \tag{3}$$

where $\text{LR}_{ku}^{(n)}$ is given by Eq. (2).

Using the likelihood ratio LR_{ku} to decide whether the data have originated from a known or an unknown hypothesis can be motivated based on statistical considerations such as standard frequentist likelihood ratio tests and importantly also Bayesian inference where LR_{ku} corresponds to a special case of a Bayes factor (Jeffreys, 1961; Wagenmakers et al., 2018). Specifically, LR_{ku} corresponds to the prescriptive factor by which a rational agent should update their beliefs about whether the data originate from a known or unknown hypothesis after having observed the data at hand.

Our use of the likelihood ratio LR_{ku} can also be motivated based on its close resemblance to one of the most successful cognitive models for recognition memory from psychological research. A typical recognition memory experiment features a study phase in which participants study a list of words. In a subsequent test phase, participants are presented with one word at a time and they have to decide whether the presented word was part of the study list ("old" word) or not ("new" word). To infer and measure the cognitive processes underlying performance in such a task, Shiffrin and Steyvers (1997) proposed the seminal *retrieving effectively from memory* (REM) model. This model aims to describe the sequence of cognitive steps that takes place when participants decide whether a presented word is "old" or "new". The key quantity that is used to decide whether a word is "old" or "new" is a likelihood ratio which contrasts the probability of the observed data given the presented word is "old" against the probability of the observed data given the presented word is "new". When one replaces "old" and "new" with "known" and "unknown", the intimate connection of LR_{ku} with this cognitive model for recognition memory becomes apparent. In our case, the task is to recognise whether the observed data have originated from a known or unknown hypothesis as opposed to recognising whether a presented word has been part of the study list or not.

5.2. Simulation study

We tested the proposed algorithm on the three scenarios described above. Specifically, in Scenario 1, the bot's candidate set of hypotheses included all one-feature hypotheses, in Scenario 2, the bot's candidate set of hypotheses included all two-feature hypotheses, and in Scenario 3, the bot's candidate set of hypotheses included all hypotheses

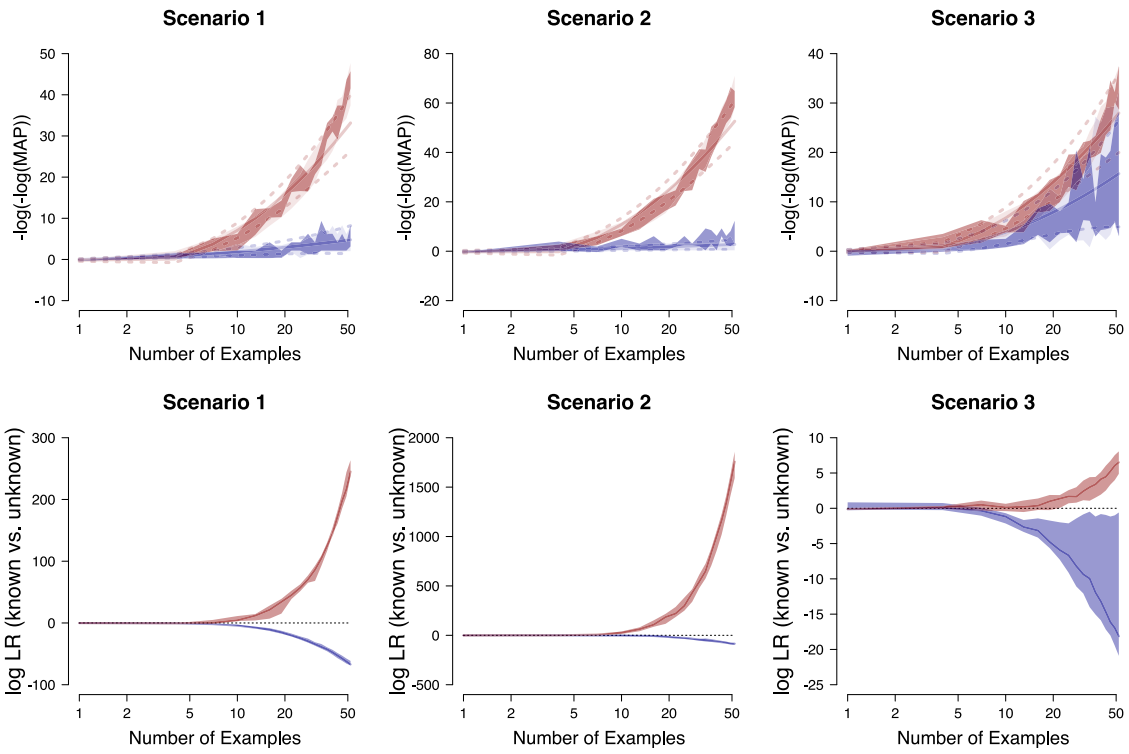


Fig. 4. The first row displays in light colors as shaded areas the .25 and .75 quantiles of the "known" and "unknown" MAP trajectories for the training phase ("known" in red, "unknown" in blue). The regression fit is displayed as a solid line, accompanied by the .25 and .75 quantiles as dotted lines. The first row also displays in darker colors as shaded areas the .25 and .75 quantiles of the "known" and "unknown" MAP trajectories for the application phase. The second row displays the log likelihood ratio from Eq. (2) in favor of "known" vs. "unknown" (median accompanied by .25 and .75 quantiles as shaded area).

in $\mathcal{P}_1$. We then applied the algorithm described above. The number of observations was set to 52 both in the training phase and the application phase and for the training phase, we repeated the process of leaving one hypothesis out 30 times. For each of these 30 repetitions, we generated 30 data sets for each of the hypotheses. Fig. 4 displays the results. The first row displays in light colors as shaded areas the .25 and .75 quantiles of the "known" and "unknown" MAP trajectories for the training phase ("known" in red, "unknown" in blue). The regression fit is displayed as a solid line, accompanied by the .25 and .75 quantiles as dotted lines. It is apparent that the location-scale regression models provide a satisfactory fit in all scenarios. The first row also displays in darker colors as shaded areas the .25 and .75 quantiles of the "known" and "unknown" MAP trajectories for the application phase. These look similar to the training ones and the regression fit, except for Scenario 3. The second row displays the log likelihood ratio from Eq. (2) in favor of "known" vs. "unknown" (median accompanied by .25 and .75 quantiles as shaded area). As more observations arrive, this log likelihood ratio accurately distinguishes between known and unknown hypotheses for all three scenarios. In terms of absolute strength of evidence provided by the computed likelihood ratio, inspecting the y-axis labels reveals that there appears to be more evidence in favor of the true data-generating hypothesis being either "known" or "unknown" in the first two scenarios than in Scenario 3.

6. Discussion

Awareness of another agent's goals is not only important in human interactions, but also in human–bot settings. A bot being aware of the fact that it does not understand the human's goals is crucial not only in terms of maximizing efficiency, but can be a matter of life and death in scenarios such as self-driving vehicles that interact with humans in traffic. In this article, we have proposed a metacognitive decision process which allows the bot to stop lower-level decision processes once sufficient evidence has accumulated that it does not understand the human's goals. The proposed approach is based on monitoring the maximum a posteriori probability (MAP) of the considered hypothesis set. We have also proposed a concrete algorithm that enables a bot to apply this approach in practice.

6.1. Conceptual considerations

Overall, our simulation studies demonstrate that considering the MAP as a function of the number of observations can be a useful tool for assessing whether the true hypothesis is in the considered set. Specifically, the MAP converged quicker to 1 when the true hypothesis was in the set. Scenario 3 suggests that considering the MAP as a function of the number of observations could potentially be challenging in certain cases. This may be partially due to the specific example we chose. For instance the observed data are discrete and some models may be equally far away from the true model in a KL-divergence sense, etc. However, this may also occur in other scenarios, for instance, when the true hypothesis is not in the considered set, but one hypothesis in the set is much closer to the true one compared to the other ones. In this case, the MAP may quickly converge to 1 to select the closest but incorrect model. This highlights that posterior probabilities are a relative measure of evidence and their behavior depends on all considered hypotheses. Consequently, our proposed approach is not going to be a perfect solution in all scenarios. However, we believe it can be a useful step towards enabling bots to realise that they don't know the goals of the human they are interacting with which is crucial for improving bots' performance. Endowing bots with a mechanism for realising that they don't know is also a timely topic due to the advent of powerful AI chatbots such as ChatGPT created by OpenAI (https://chat.openai.com/chat). Specifically, one commonly reported issue with ChatGPT is that it can provide incorrect answers with high confidence (Steyvers et al., 2024), that is, it does not seem to have an appropriate mechanism in place for knowing when it does not know.

To apply our proposed algorithm in practice, the bot needs to have available a set of hypotheses that can be used to simulate data and that can be tested against observed data. In our simulation study, we have demonstrated that our algorithm performs well in scenarios where the bot's hypothesis space is similar to the human's hypothesis space (e.g., the bot only misses one feature). However, in real life, the bot might be missing many of the features that are relevant to the human's true data-generating hypothesis. To a degree, this may simply be a function of the bot not being equipped with the relevant sensors. For instance, suppose the human behaves differently when the weather is hot compared to when it is cold. If the bot is not equipped with a temperature sensor, it is impossible for the bot to include the correct data-generating hypothesis in its considered set. This concern can be addressed to a degree by the designers of the bot carefully considering what sensors may be relevant to the bot's tasks, and also equipping the bot with a sensible set of hypotheses to work with. We still hope that even in the scenario where the bot misses many relevant features, our algorithm would simply indicate "don't know", such that the conclusion would be to ask for assistance. However, how well our algorithm performs in these scenarios is an empirical question that is beyond the scope of this article.

Another conceptual challenge is that the bot's sensors may be noisy. Specifically, suppose that in our example, the human presents a red triangle with a "9" written on it, but the bot mistakenly registers the object as a red triangle with a "6" on it. In our simulation study, we have explicitly accounted for variability (noise) on the human's side via the likelihood function, to take into account that the human does not always present an object in line with the rule. In a similar fashion, one could augment the likelihood function to take into account noise on the bot side. This may take the form of a free parameter that is estimated from data, or, if the noise level of the sensors is known, for instance, from extensive tests in the factory, one could simply incorporate this known variability into the likelihood function.

6.2. Computational considerations

Our proposed algorithm assumes that the bot is able to conduct computations based on all hypotheses in the considered set. This could be practically challenging due to a number of reasons. First, the number of hypotheses in the set could be very large. In this case, in the training phase, it may not be feasible to repeatedly simulate data from all hypotheses in the set. However, in this case one may approximate this process by, in each iteration of the algorithm, randomly selecting a subset of the hypotheses such that simulating data becomes feasible. In the application phase, it is undesirable to reduce the computational burden by randomly subsetting the hypothesis space since this could result in disregarding relevant hypotheses. Instead, one may apply a more principled approach for reducing the hypothesis space, such as "Occam's window" (Madigan & Raftery, 1994). This approach reduces the computational burden by conducting a sequence of pairwise comparisons of nested hypotheses. If the simpler hypothesis is preferred, the more complex hypothesis is removed from the hypothesis space. If the more complex hypothesis is preferred by a large margin, the simpler hypothesis, and all hypotheses nested within this simpler hypothesis, are removed from the hypothesis space. Only hypotheses between these two extremes, the ones that fall in "Occam's window", are retained for inference.

Second, even when the number of hypotheses is manageable, simulating from a hypothesis could take time. However, one advantage of the proposed algorithm is that the computationally intense training phase need not happen while real data is observed, but can be conducted and saved in advance or can be achieved during downtime. For instance, consider the case of a social bot assisting a human in their daily tasks such as setting the table, cleaning up, etc. If simulating hypotheses in real time or a priori is challenging, this could be achieved during times of inactivity, such as when the human sleeps or performs other activities. Furthermore, since there may be many social bots assisting different humans, it may be possible to share experiences with humans and hypothesis simulations across bots via online functionality.

Another computational consideration is that calculating the MAP requires the bot to evaluate the marginal likelihood, the probability of the data given a hypothesis, of all hypotheses in the set. Depending on the complexity of the considered hypotheses, this may not be possible analytically such that simulation-based approaches such as bridge sampling (Gronau et al., 2017; Meng & Wong, 1996) or path sampling (Gelman & Meng, 1998) are required. Depending on the considered hypotheses and observed data, computing the marginal likelihood using these approaches may be too time-consuming to achieve in real time as data arrive. However, in this case, fast approximations to the marginal likelihood may be useful, such as variational Bayesian inference (Galdo, Bahg, & Turner, 2020).

An avenue for future research is to apply the proposed algorithm in practice. Specifically, a first step would be to design an experiment in which a bot assists a human and then investigate whether, using the proposed algorithm, the bot can correctly identify whether it does not know the human's goals. The ultimate goal would be to endow bots that assist humans in real life with our proposed metacognitive decision algorithm and investigate whether it increases their performance.

7. Conclusion

Knowing when one does not know something is crucial for both humans and bots for stopping the execution of inappropriate behavior and, ideally, attempting to learn more appropriate actions. From a statistical point of view, this can be translated to assessing whether none of the hypotheses in a considered set is correct. Here we investigated a novel approach for making this assessment based on monitoring the maximum a posteriori probability (MAP) of a set of candidate hypotheses as new observations arrive. In simulation studies, we demonstrated that this can be a useful tool, but we also cautioned that there may be cases where this is more challenging. In future work we plan to explore how well this approach works in practice when bots assists humans to achieve their goals.

CRediT authorship contribution statement

Quentin F. Gronau: Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft. **Mark Steyvers:** Conceptualization, Formal analysis, Methodology, Writing – review & editing. **Scott D. Brown:** Conceptualization, Formal analysis, Methodology, Writing – review & editing.

Declaration of competing interest

none

Data availability

No data have been used for this study. Code for reproducing the analyses can be found at: https://osf.io/t4n63/.

Acknowledgments

This work was partially supported by the Australian Research Council (DP210100313) and the Australian Defence Science and Technology Group (AUSMURIIV000001).

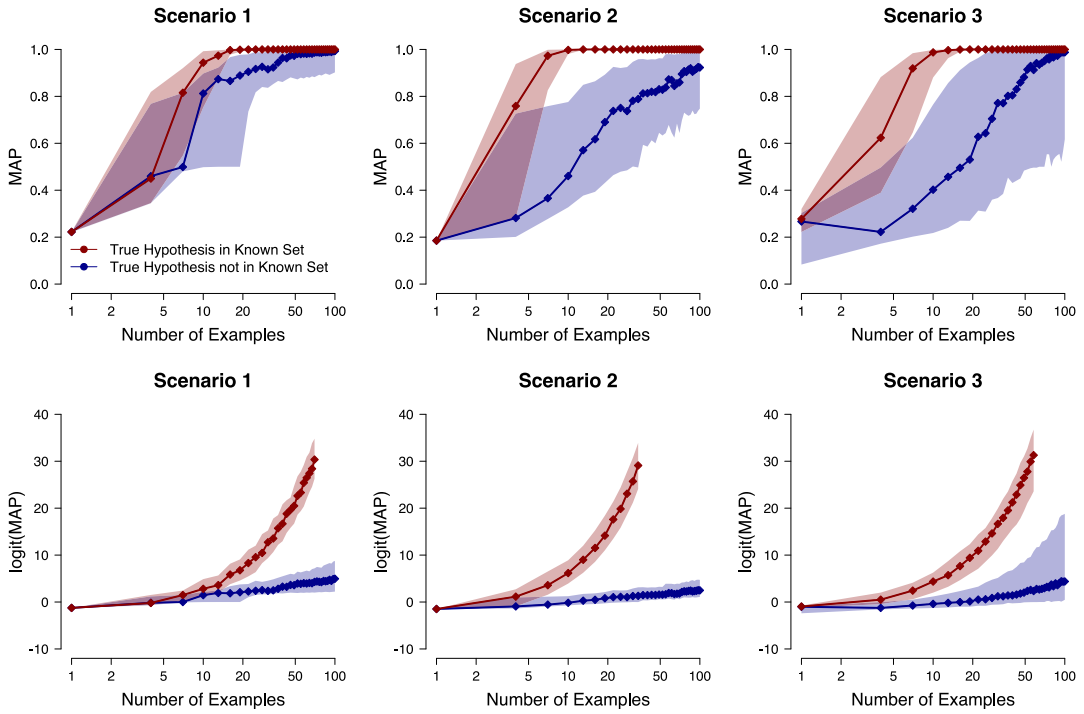


Fig. A.1. Simulation results for p_c freely estimated. The upper row displays, for each scenario separately, the median of the MAP (dots), accompanied by the 25% and 75% quantiles (shaded area), as a function of the number of presented examples (on a log scale), split by whether the true hypothesis was in the considered set (red symbols) or not (blue symbols). The bottom row displays the logit transformed MAP as a function of the number of presented examples in a similar manner.

Appendix. Simulation results for p_c freely estimated

Fig. A.1 displays the results of a simulation study similar to the one reported in the main text that only differed in that the probability of setting the feature(s) of interest to their correct values, p_c, was a free parameter and not fixed. We assigned p_c a uniform prior distribution on the interval $(0, 1)$. As is apparent from Fig. A.1, the results are very similar to the ones for the case where p_c was fixed to 0.7 (i.e., Fig. 3). A small difference is that for the most challenging case, Scenario 3, the separation in the MAP trajectories between unknown and known data-generating hypotheses is a little better.

References

Bobu, A., Bajcsy, A., Fisac, J. F., Deglurkar, S., & Dragan, A. D. (2020). Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, *36*, 835–854.

Boden, M. A. (2009). Computer models of creativity. *AI Magazine*, *30*(3), 23–23.

Chatterjee, D., Maitra, T., & Bhattacharya, S. (2020). A short note on almost sure convergence of Bayes factors in the general set-up. *The American Statistician*, *74*, 17–20.

Chu, Y., & MacGregor, J. N. (2011). Human performance on insight problem solving: A review. *The Journal of Problem Solving*, *3*(2), 119–150.

Fridovich-Keil, D., Bajcsy, A., Fisac, J. F., Herbert, S. L., Wang, S., Dragan, A. D., et al. (2020). Confidence-aware motion prediction for real-time collision avoidance. *International Journal of Robotics Research*, *39*, 250–265.

Galdo, M., Bahg, G., & Turner, B. M. (2020). Variational Bayesian methods for cognitive science. *Psychological Methods*, *25*(5), 535–559.

Gelman, A., & Meng, X.-L. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, *13*, 163–185.

Glucksberg, S., & McCloskey, M. (1981). Decisions about ignorance: Knowing that you don't know. *Journal of Experimental Psychology: Human Learning and Memory*, *7*(5), 311.

Golden, R. M., Henley, S. S., White, H., & Kashner, T. M. (2016). Generalized information matrix tests for detecting model misspecification. *Econometrics*, *4*, 46.

Gronau, Q. F., Sarafoglou, A., Matzke, D., Ly, A., Boehm, U., Marsman, M., et al. (2017). A tutorial on bridge sampling. *Journal of Mathematical Psychology*, *81*, 80–97, Retrieved from http://dx.doi.org/10.1016/j.jmp.2017.09.005.

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford, UK: Oxford University Press.

Krauz, O., & Tabrikian, J. (2019). Detection of modeling misspecification using cross-entropy test. In *2019 IEEE 8th international workshop on computational advances in multi-sensor adaptive processing* (pp. 520–524). IEEE.

Kumar, A., & Steyvers, M. (2022). Help me help you: A computational model for goal inference and action planning. http://dx.doi.org/10.31234/osf.io/f6gm3.

Madigan, D., & Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, *89*, 1535–1546.

McGreggor, K., Kunda, M., & Goel, A. K. (2011). Fractal analogies: Preliminary results from the raven's test of intelligence. In *ICCC* (pp. 69–71).

Meng, X.-L., & Wong, W. H. (1996). Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, *6*, 831–860.

Nelson, T. O. (1990). Metamemory: A theoretical framework and new findings. In *Psychology of learning and motivation: vol. 26*, (pp. 125–173). Elsevier.

Presnell, B., & Boos, D. D. (2004). The IOS test for model misspecification. *Journal of the American Statistical Association*, *99*, 216–227.

Puig, X., Shu, T., Li, S., Wang, Z., Liao, Y.-H., Tenenbaum, J. B., et al. (2020). Watch-and-help: A challenge for social perception and human-AI collaboration. http://dx.doi.org/10.48550/ARXIV.2010.09890, arXiv. Retrieved from https://arxiv.org/abs/2010.09890.

Shiffrin, R. M., & Steyvers, M. (1997). A model for recognition memory: REM—Retrieving effectively from memory. *Psychonomic Bulletin & Review*, *4*, 145–166.

Steyvers, M., Tejeda, H., Kumar, A., Belem, C., Karny, S., Hu, X., et al. (2024). The calibration gap between model and human confidence in large language models. arXiv preprint arXiv:2401.13835.

Tenenbaum, J. B., & Griffiths, T. L. (2001). Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, *24*(4), 629–640.

Wagenmakers, E.-J., Marsman, M., Jamil, T., Ly, A., Verhagen, A. J., Love, J., et al. (2018). Bayesian statistical inference for psychological science. Part I: Theoretical advantages and practical ramifications. *Psychonomic Bulletin & Review*, *25*, 35–57.

Yang, J., Zhou, K., Li, Y., & Liu, Z. (2021). Generalized out-of-distribution detection: A survey. arXiv preprint arXiv:2110.11334.