

Meta-learning ecological priors from large language models explains human learning and decision making

Akshay K. Jagadish^{1,2, 3, 4, *}, Mirko Thalmann¹, Julian Coda-Forno¹, Marcel Binz^{1,+}, and Eric Schulz^{1,+}

¹Institute for Human-Centered AI, Helmholtz Computational Health Center, Munich, Germany

²Computational Principles of Intelligence, Max Planck Institute for Biological Cybernetics, Tübingen, Germany

³Eberhard Karls University of Tübingen, Tübingen, Germany

⁴Princeton AI Lab, Princeton University, Princeton, USA

*corresponding author: akshaykjagadish@gmail.com

+these authors contributed equally to this work

ABSTRACT

Human cognition is profoundly shaped by the environments in which it unfolds. Yet, it remains an open question whether learning and decision making can be explained as a principled adaptation to the statistical structure of real-world tasks. We introduce ecologically rational analysis, a computational framework that unifies the normative foundations of rational analysis with ecological grounding. Leveraging large language models to generate ecologically valid cognitive tasks at scale, and using meta-learning to derive rational models optimized for these environments, we develop a new class of learning algorithms: Ecologically Rational Meta-learned Inference (ERMI). ERMI internalizes the statistical regularities of naturalistic problem spaces and adapts flexibly to novel situations, without requiring hand-crafted heuristics or explicit parameter updates. We show that ERMI captures human behavior across 15 experiments spanning function learning, category learning, and decision making, outperforming several established cognitive models in trial-by-trial prediction. Our results suggest that much of human cognition may reflect adaptive alignment to the ecological structure of the problems we encounter in everyday life.

Significance Statement

Humans are remarkably adaptive, making good decisions in complex, uncertain environments. But where do these abilities come from and how can we model them? This work introduces a new approach that combines insights from psychology and machine learning to explain human cognition as an adaptation to ecological environments. By using large language models to generate realistic problems and training neural networks to solve them, we show that simple general-purpose systems can mirror how people learn, categorize, and decide. Our results suggest that much of human learning and decision making may be explained by attunement to the structure of the world around us.

Introduction

It is a truth universally acknowledged that a mind in search of a decision is influenced by its environment. Charles Darwin¹ showed that species are adapted to their environmental niche to survive. Egon Brunswik² proposed that people carefully interpret the signals in their surroundings to make judicious decisions. Herbert Simon³ emphasized that human behavior is the result of the interplay between limited cognitive resources and the structure of the environment. Gerd Gigerenzer⁴ furthered this notion by introducing the concept of ecological rationality, proposing that minds adapt to their environments by relying on simple context-specific strategies. Yet it remains unclear how attuned human learning and decision making are to the statistical structure of ecologically valid environments.

Two prominent frameworks have sought to address this question through computational modeling: rational analysis⁵ and ecological rationality⁶. While rational analysis seeks optimal strategies within formal models of the environment, ecological rationality emphasizes heuristics tuned to the structure of real-world tasks. Although rational analysis offers a principled way to derive an adaptive strategy, it requires defining a formal model of the environment. This requirement limits its applicability to relatively simple environments. Ecological rationality, on the contrary, offers a flexible way to model real-world behavior, but it relies on the researcher to hand-design suitable heuristics. This reliance makes it challenging to extend the framework to new domains where effective heuristics have yet to be discerned.

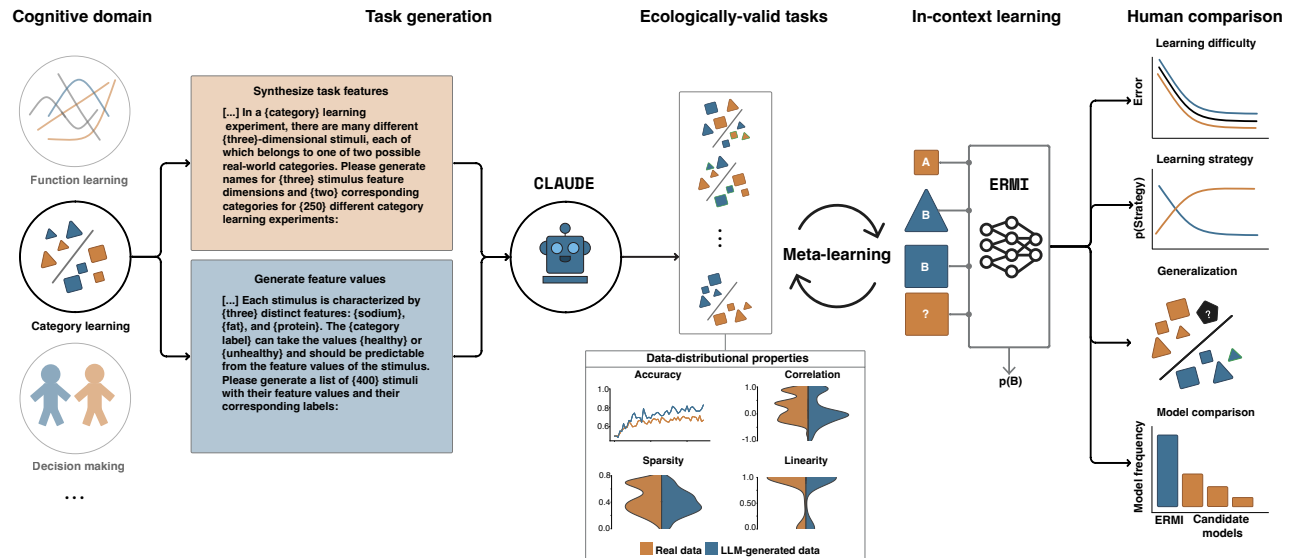


Figure 1. Schematic of Ecologically Rational Meta-learned Inference: Ecologically Rational Meta-learned Inference (ERMI) is domain-agnostic and can be applied to any cognitive domain. Let us consider category learning as the domain of interest for this illustration. The first step in deriving ERMI is to use a LLM (e.g., CLAUDE-V2) to generate ecologically valid tasks. Task generation from an LLM proceeds in two stages: first, the LLM synthesizes plausible task features (e.g., predicting whether a food item is healthy or unhealthy based on sodium, fat, and protein content); second, it generates corresponding input-target pairs consistent with these features¹⁰. Once a sufficient number of category learning tasks are generated, we analyze their distributional properties, such as classification accuracy, input feature correlation, sparsity in predictive features, and linearity of category structures, and compare them to real-world datasets (e.g., OpenML-CC18¹⁴) to verify their ecological validity. We then derive computational models that internalize these ecological priors by training a neural network (e.g., transformer¹⁵) on the LLM-generated tasks using meta-learning. This yields a family of in-context learners, termed Ecologically Rational Meta-learned Inference (ERMI), which flexibly adapt to the statistical structure of naturalistic problems. In category learning, the resulting models are evaluated against human behavior across four key dimensions: learning difficulty, learning strategy, generalization, and quantitative fit to human behavior through model comparison.

We introduce ecologically rational analysis, a framework that synthesizes the strengths of rational analysis and ecological rationality. This framework enables the automated derivation of computational models that implement approximately optimal strategies directly adapted to the statistical structure of natural environments. These models can subsequently be interrogated – through classical psychological experiments, for example – to elucidate how and which environmental properties give rise to human behavior.

To develop this framework, we draw on two recent advances in machine learning: large language models (LLMs) and in-context learning⁷. LLMs are generative models trained on internet-scale corpora, capable of capturing the statistical regularities that characterize real-world tasks and domains^{8,9}. We harness this capacity to generate ecologically valid learning environments: problems that approximate the kinds of structure humans are likely to encounter in everyday life^{10,11}. In-context learning refers to the ability of neural networks to learn from examples presented within a sequence, adapting their behavior purely through internal activations, without any parameter updates⁷. When derived via meta-learning, in-context learning has been shown to approximate Bayes-optimal inference conditioned on the statistics of its training distribution^{12,13}.

By meta-learning on tasks generated by LLMs, we develop models that internalize the ecological priors inherent in these environments. We term this class of models Ecologically Rational Meta-learned Inference (ERMI): a family of in-context learners that flexibly adapt to the statistical structure of naturalistic problems. We find that ERMI robustly captures human behavior across 15 experiments encompassing three core domains of cognition: function learning, category learning, and decision making. Beyond accounting for hallmark behavioral signatures within each domain, ERMI yields superior trial-by-trial predictions of human choices relative to a diverse array of established cognitive models. Collectively, these findings suggest that adaptive alignment with environmental statistics is sufficient to account for a broad spectrum of human learning and decision making behavior.

Results

In what follows, we describe ERMI and demonstrate how it can be used to model human learning and decision making across diverse cognitive domains; see Figure 1 for an overview.

The core idea behind ERMI is that adaptive behavior reflects the internalization of ecological priors. To capture these priors, ERMI uses LLMs as generative engines to construct ecologically valid tasks (see scalable generation of cognitive tasks from LLMs in Methods). This generation process involves two stages: first, the LLM proposes plausible task features (e.g., predicting weight from calories consumed); second, it generates corresponding input-target pairs for the given task features (e.g., specific calorie and weight values)¹⁰. Importantly, the generated targets correspond to ground truth values and *not* human predictions. By querying LLMs, we synthesize a rich and diverse set of cognitive tasks that approximate the distribution of problems found in natural settings.

ERMI then uses meta-learning^{16–18} to derive computational models adapted to the LLM-generated tasks (see Methods). The resulting models implement approximately Bayes-optimal policies that adapt in-context – modifying internal activations rather than parameters – to the structure of encountered problems^{12,13}. This approach allows us to systematically test whether optimal alignment with ecological statistics is sufficient to account for hallmark patterns of human learning and decision making.

In a series of experiments, we demonstrate how ERMI captures human behavior across three core domains of cognition: function learning, category learning, and decision making. For each domain, we show that it replicates core behavioral signatures and provides better trial-by-trial predictions of human choices compared to established cognitive models.

Function learning

Psychologists have been interested in understanding how people learn the functions underlying the association between an input and a target since the 1960s¹⁹. Much of these studies have focused on mapping a single-dimensional input to a response, called single-cue function learning^{20,21}, which is also the focus of this work.

In these tasks, participants observe input-output pairs, typically receiving feedback on the true response after each prediction. The underlying function is unknown and must be inferred through trial and error. Once trained, participants are tested on previously unobserved inputs within the range of their prior experience (interpolation) or outside that range (extrapolation). Previous work has revealed several hallmark findings: people interpolate more accurately than they extrapolate, sometimes performing as well on novel interpolated inputs as on the training set itself²²; and they exhibit systematic biases when generalizing, favoring linear functions with positive slopes and minimal offsets^{21,23,24}.

ERMI provides a framework for probing the origins of these behavioral patterns. Following the procedure outlined previously, we first construct a dataset of about 10,000 one-dimensional function learning tasks designed to reflect the diversity of functional relationships found in natural environments (see Figure 2C for examples). We analyze the statistical properties of the LLM-generated tasks and compare the tasks with real-world regression problems²⁵. We found that the generated functions comprised approximately 75% linear, 12% exponential, 7% quadratic, and 6% periodic relationships (see Figure 2A), a distribution that mirrors both environmental regularities and human difficulty rankings in function learning²⁶. A model-based analysis of linear fits revealed a pronounced bias toward positive slopes with near-zero offsets (see Figure 2B), consistent with known human biases in extrapolation^{27,28}.

Then, we asked to what extent ERMI can replicate the characteristic patterns of human function learning? Drawing on prior work²⁶, we focused on five well-established findings in function learning: (i) linearly increasing functions are learned more readily than decreasing functions^{30–32}; (ii) linearly increasing functions are also learned faster than nonlinearly increasing functions^{22,33}; (iii) monotonic functions are learned more effectively than non-monotonic ones^{19,30,33}; (iv) cyclic functions are more challenging than non-cyclic functions³³; and (v) generalization is more accurate in interpolation than in extrapolation^{19,22,34}.

To test whether ERMI reproduces these patterns, we sampled functions from each class and assessed the model's learning dynamics. We measured learning speed and accuracy using the rate of change and mean-squared error (MSE) across trials (see Methods for details). Strikingly, ERMI mirrored human behavior across all five phenomena: (i) it learned positive linear functions ($M_{\text{MSE}}=0.5260$, $\text{SEM}=0.0043$, $t=-70.1587$, $p<0.001$) better than negative linear functions ($M_{\text{MSE}}=0.9339$, $\text{SEM}=0.0039$); (ii) it grasps linear functions with positive slopes more rapidly, reaching minimum MSE in fewer trials ($M_{\text{trial}}=13.29$, $\text{SEM}=0.051$) compared those with negative slopes ($M_{\text{trial}}=14.22$, $\text{SEM}=0.048$; $t=-13.38$, $p<0.01$); (iii) it mastered monotonic functions ($M_{\text{MSE}}=0.8895$, $\text{SEM}=0.0028$, $t=-145.5417$, $p<0.01$) more accurately than non-monotonic ones ($M_{\text{MSE}}=1.5255$, $\text{SEM}=0.0032$); (iv) it learned non-cyclic functions ($M_{\text{MSE}}=1.0423$, $\text{SEM}=0.0025$, $t=-89.6918$, $p<0.01$) more readily than cyclic ones ($M_{\text{MSE}}=1.5508$, $\text{SEM}=0.0054$, $t=-89.6918$, $p<0.01$); and (v) it achieved better prediction performance during interpolation ($\text{MSE}=0.0017^{27}$) than during extrapolation ($\text{MSE}=0.0022$); see Figure 2D-E.

A well-established finding in the function learning literature is that humans tend to underestimate functional relationships during extrapolation, particularly for linear functions, with a characteristic bias toward zero offset²⁷. To assess whether ERMI exhibits a similar pattern, we conditioned the model on input-target pairs sampled from a linear function, following the

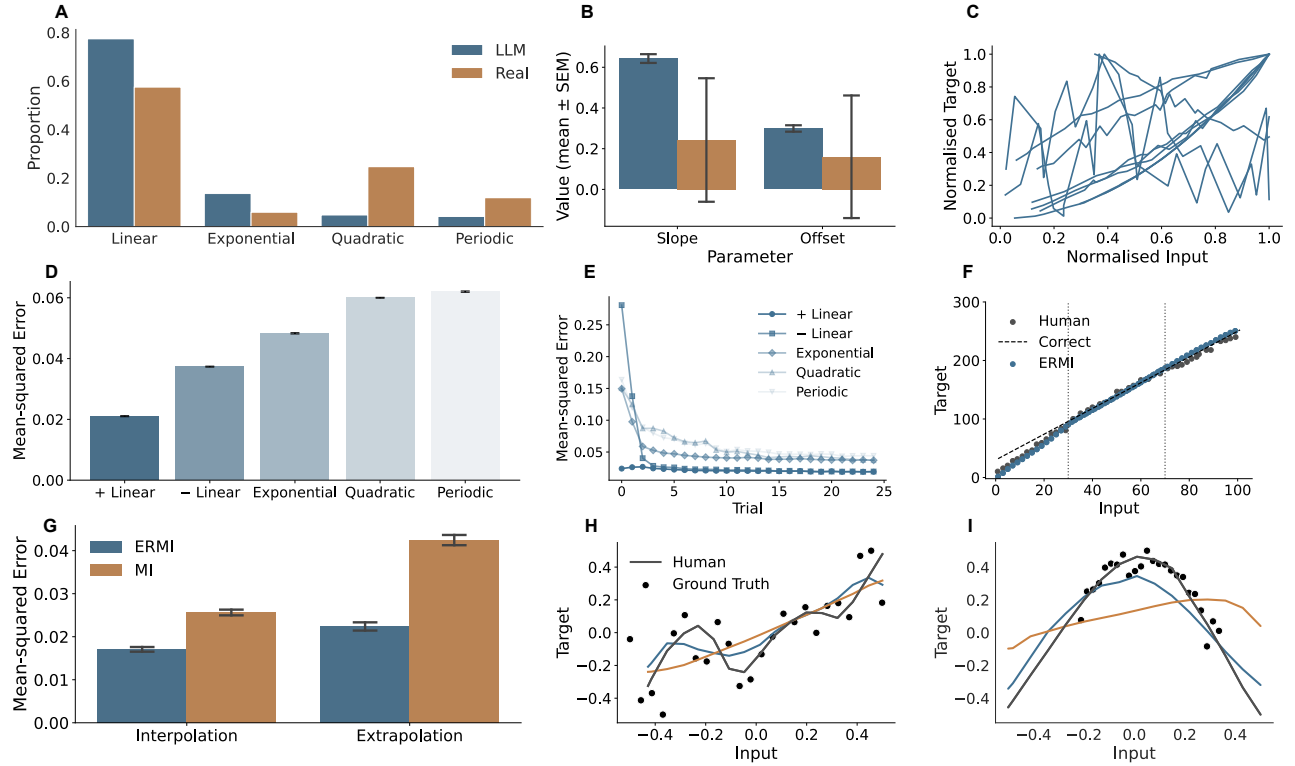


Figure 2. Function Learning: **A** Proportions of different function types in real-world datasets²⁵ and LLM-generated datasets. **B** Parameters for slope and offset for linear functions fitted to both datasets. **C** Example functions sampled from the LLM-generated datasets. **D** Mean-squared error (MSE) of ERMI when simulated on five function types, namely, positively-sloped linear, negatively-sloped linear, exponential, quadratic, and periodic functions (mean over all runs and trials). **E** MSE of ERMI for the five function types mentioned above unrolled over trials (mean over all runs). **F** Simulations of ERMI on linear function from²⁷ along with human predictions extracted from the original plot (gray lines mark interpolation range between 30 to 70 and extrapolation region between 0 to 30 and between 70 to 100). **G** MSE during interpolation (Experiment 2 of original study) and extrapolation (Experiment 4) for ERMI and Meta-learned inference (MI) with hand crafted prior when simulated on tasks from²⁹. **H** Representative example for interpolation. **I** Representative example for extrapolation.

procedure of Kwantes and Neal²⁷ (see human studies in SI for additional details), and simulated its predictions for input values outside the training range. We compared ERMI's responses to human data in both interpolation and extrapolation regimes. For human responses, we drew on data from Experiment 2 of Kwantes and Neal²⁷, in which participants directly estimated numerical values for a given input – an evaluation method that closely parallels our procedure for ERMI. We found that, like humans, ERMI systematically underestimated responses in the extrapolation range, with a stronger bias in the lower region (MSE=0.0043; 0-30 on the x-axis, which is marked using grey lines in the Figure 2F) than in the upper region (MSE=0.0003; 70-100 on the x-axis). It showed a bias towards zero offset comparable to that observed in human participants (see Figure 2F). In addition, ERMI's predictions agreed better (MSE=0.0002) with human responses than a meta-learned (MI; MSE=0.00054) with hand-crafted prior used by Lucas and colleagues³⁵; see Methods for details.

Beyond qualitative signatures, a critical test of any model is whether it can capture the fine-grained structure of human behavior at the trial level. To this end, we evaluated ERMI on the function estimation task introduced by Little and colleagues²⁹. In this task, participants viewed 24 scatter plots, each depicting data from a different fictional scientific experiment, and were asked to draw what they believed to be the true underlying causal function. The scatter plots were generated from linear, quadratic, or cubic polynomial functions with added Gaussian noise. Crucially, the scatter plots were presented in two distinct formats. In the zoomed-in condition, data points filled the entire plotting area, enabling assessment of interpolation. In the zoomed-out condition, data points were centrally located and occupied only 40% of the plot area, encouraging participants to extrapolate beyond the range of the observed data; see human studies in the SI for additional details.

We conditioned ERMI on the same training data shown to participants and generated predictions for the identical input values. To quantify model fit, we computed the MSE between ERMI's predictions and human responses across all test inputs. As shown in Figure 2G, ERMI provided a closer fit to human judgments than MI in both interpolation and extrapolation conditions. Specifically, ERMI achieved a lower MSE ($M_{\text{MSE}}=0.0171$, $\text{SEM}=0.0014$, $t = -9.9944$, $p < 0.01$) compared to MI ($M_{\text{MSE}}=0.0256$, $\text{SEM}=0.0020$) during interpolation, and likewise outperformed MI during extrapolation (ERMI: $M_{\text{MSE}}=0.0223$, $\text{SEM}=0.0018$; MI: $M_{\text{MSE}}=0.0424$, $\text{SEM}=0.0034$, $t = -13.1479$, $p < 0.01$). For illustrative examples, Figure 2H-I shows ERMI's predictions, alongside those of MI, for interpolation and extrapolation condition from a representative participant; see Figure S2 in the SI for additional examples. Together, these findings suggest that a rational model attuned to the statistics of ecologically valid function learning problems is sufficient to capture much of human function learning.

Category learning

To examine whether these findings generalize, we turned to a second domain that has been widely studied in cognitive psychology, namely category learning³⁶. In a typical category learning task, participants are presented with inputs sequentially and must assign each one to a set of known categories. After each trial, they receive feedback that indicates whether their classification was correct. The underlying rule that governs category membership – referred to as the category structure – is hidden from participants and must be inferred through trial and error. During the test phase, participants classify both previously seen (training) and novel (transfer) input without receiving feedback. This design allows for simultaneous assessment of learning performance on familiar examples and generalization to new instances.

It has been observed that humans find learning certain category structures more difficult than others³⁷. Furthermore, the categorization strategy they use changes during the course of the experiment from an exemplar-based strategy to a prototype-based strategy³⁸. In addition, the way they generalize to unseen inputs is systematic, following a rule-plus-exception-based model³⁹.

To investigate the role of ecological adaptation in explaining these findings, we again turn to ecologically rational analysis. Following a previously established procedure, we generated about 10,000 category learning problems and inspected their underlying statistics. Like in function learning, we found that LLM-generated category learning problems capture key statistical properties of real-world classification datasets¹⁴. Specifically, we observe that (i) the generated category learning problems are noisy, yet classifiable, like real-world classification problems (Figure 3A); (ii) they contain inputs whose features exhibit a full range of correlations, from non-existent or low values to nearly complete correlation, as in real-world tasks (Figure 3B); (iii) only a few feature dimensions within each task substantially contribute to classification, indicating sparsity in the feature space commonplace in real-world tasks (Figure 3B; larger Gini coefficient values indicate higher sparsity); and (iv) the category structure observed in the generated classification problems is predominantly linear akin to real-world tasks (Figure 3B; higher values indicate more linearity). These findings confirm the ecological validity of LLM-generated category learning tasks.

Moving to the next step, we derived ERMI by meta-learning on these category learning problems and evaluated how well it can capture various aspects of human category learning. To explain human learning difficulties during category learning using ERMI, we consider the study by Shepard et al.³⁷. In this study, the authors considered six different category structures (labeled TYPE 1 to TYPE 6). In TYPE 1 problems, all items in a category share one particular feature value (e.g., they are all black), TYPE 2 problems are defined by a combination of two feature values (i.e., XOR problems), TYPE 3-5 problems combine a rule with exceptions, and TYPE 6 problems require the memorization of individual items as rules and the similarity to other

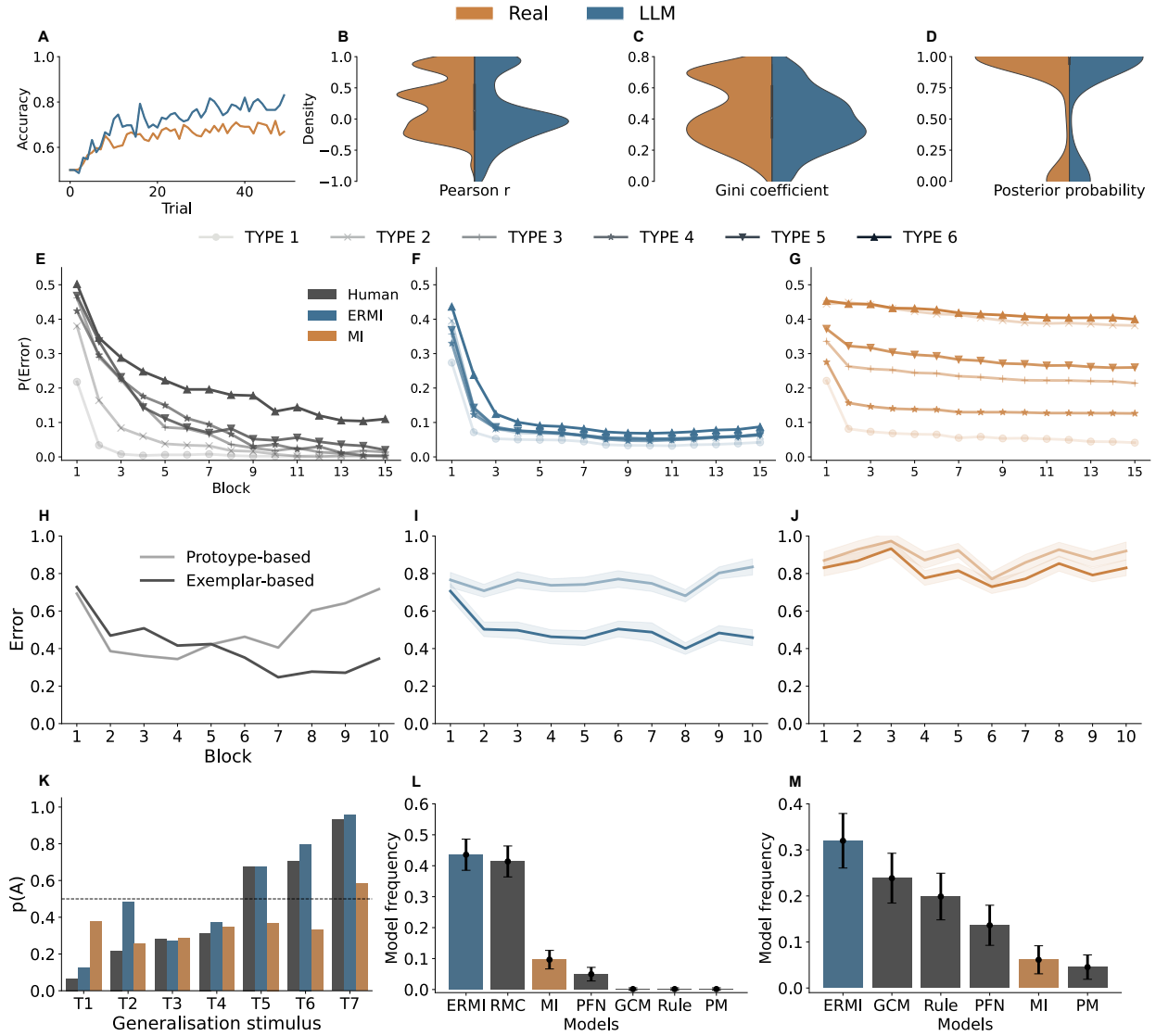


Figure 3. Category learning: **A** Mean task performance of a logistic regression model over trials for real-world classification tasks¹⁴ in orange and LLM-generated tasks in blue. **B** Density plot of Pearson's correlation coefficients between feature dimensions. **C** Gini coefficients over logistic regression weights, which provides a measure of sparsity (high values indicates greater sparsity). **D** Posterior probability measuring the linearity of category learning tasks. **E–G** Average error probabilities for each task TYPE across 16-trial blocks for **E** humans (in gray), **F** ERMI (in blue), and **G** MI (in orange). Human data in **E** reproduced from Table 1 in⁴⁰. ERMI and MI were simulated on TYPE 1–6 tasks for 50 runs using the inverse temperature (β) that minimized mean-squared error with respect to human data: $\beta = 0.4$ for ERMI and $\beta = 0.9$ for MI. **H–J** Average error of exemplar- and prototype-based models fitted to **H** human choices, **I** simulated choices from ERMI, and **J** simulated choices from MI across 56-trial blocks. Human data in **H** reproduced from³⁸. ERMI and MI were simulated using inverse temperature values fitted to participants' choices in⁴¹; ERMI ($M_\beta=0.09$, $SEM=0.01$) and MI ($M_\beta=0.17$, $SEM=0.02$). Shaded regions indicate standard error of the mean. **K** Average categorization probabilities of transfer inputs T1–T7 for humans (gray), ERMI (blue), and MI (orange). Human data reproduced from³⁹. ERMI and MI were simulated on the same experiment over 77 runs using the best-fitting inverse temperatures: $\beta = 0.9$ for ERMI and $\beta = 0.1$ for MI. **L** Posterior model frequency of participants' choices in⁴² across seven computational models. **M** Posterior model frequency of participants' choices in⁴¹ across six computational models.

items is not informative; see human studies in the SI for details. The task difficulty of the six problem types increases from 1 ($M_{p(\text{Error})} = 0.0201$) to 6 ($M_{p(\text{Error})} = 0.2048$), as shown in Figure 3E. The error rates for TYPE 2-5 problems fall between those of TYPE 1 and TYPE 6; see Table S3 in SI for details. ERMI – when simulated on tasks from the Shepard et al. study³⁷ – displayed learning curves that are difficulty dependent and follow the same ordering as people’s; see Figure 3F. Quantitatively, ERMI ($MSE = 0.03$) captured human learning difficulties better than meta-learned inference with a hand-crafted prior (MI; $MSE = 0.26$); see Methods for details.

We then investigated whether ERMI also captures human trial-by-trial choices during category learning by considering a replication of the original Shepard’s study³⁷ by Badham et al.⁴²; see human studies in the SI for details. We performed a Bayesian model comparison between ERMI and five other computational models, which included the rational model of categorization (RMC⁴³), a prototype model (PM⁴⁴), an exemplar model (generalized context model (GCM)⁴⁵), a rule-based model (Rule³⁴) and a meta-learned inference model with Bayesian logistic regression prior (MI) and Bayesian neural network prior (PFN⁴⁶); see Methods for details on the baseline models, as well as the model fitting and comparison procedure. We found that in terms of the posterior model frequency (PMF), which measures how often a model offers the best explanation in the population, ERMI explains human choices more frequently ($M_{PMF}=0.43$, $SEM=0.05$) compared to the other models, with the RMC coming in close second ($M_{PMF}=0.41$, $SEM=0.05$); see Figure 3L.

After that, we tested whether ERMI shows a similar shift in its categorization strategy as humans³⁸. In this study, participants classified 14 six-dimensional inputs into two categories. These categories were assigned based on a nonlinear decision rule; see human studies in the SI for details. The authors then fitted a prototype and an exemplar model to the observed behavior and found that the prototype model better explained the people in the early blocks but in the later blocks, their choices aligned more closely with the exemplar model, as shown in Figure 3H. When simulated on tasks from the same study, we found ERMI’s strategy to be indistinguishable between prototype-based and exemplar-based in the beginning of the experiment, but with experience, it became increasingly more exemplar-based as observed in humans (see Figure 3I). In contrast, MI does not display such a transition in strategy, as shown in Figure 3J. Furthermore, we compared ERMI with other competing models in the prediction of human choices at the trial level, for which we used human data from a replication of the original study by Devraj et al.⁴¹. As shown in Figure 3M ERMI ($M_{PMF}=0.32$, $SEM=0.06$) predicted human choices the most frequently, followed by the GCM ($M_{PMF}=0.24$, $SEM=0.05$) and the rule-based model ($M_{PMF}=0.20$, $SEM=0.05$).

Finally, we examined whether ERMI displays the same generalization patterns as people when they observe inputs not part of the training phase³⁹. In the training phase of this study, participants performed binary classification of nine four-dimensional inputs. Subsequently, in the test phase, they were probed on seven transfer inputs (labeled T1-T7; see Methods for their encodings) for which they did not receive any feedback; see human studies in the SI for details. The latter was intended to examine how they would generalize the learned category structure to unseen inputs; see Method for details. Figure 3K shows the proportion of responses in which the participants assigned category A to the seven transfer inputs (in gray). It can be seen that participants assigned the transfer inputs T5, T6, and T7 mainly to category A and the inputs T1, T2, T3, and T4 mainly to category B. ERMI – when evaluated on the same task – generalizes to unseen inputs in a human-like way by classifying the inputs T1, T3, and T4 more often as category B and the inputs T5, T6, and T7 more often as category A; see Figure 3I (in blue). The only deviation from human-like generalization is input T2. Although ERMI classified it as category A at the chance level, the participants predominantly assigned it to B. We speculate that this is because T2 resembles the category prototype along two only dimensions, while other inputs categorized as B matched along three dimensions. Yet again, MI did not show the same pattern as in humans, both qualitatively (Figure 3K; in orange) and quantitatively, with the Euclidean distance between the choice probabilities of humans and ERMI (0.29) being lower than between humans and MI (0.67).

These results indicate that in addition to capturing human function learning, ERMI also captures human category learning.

Decision making

The question of how people decide between multiple options and how they improve on it with experience has been extensively studied in economics⁴⁷, psychology⁴, and neuroscience⁴⁸. Does ERMI also extend to this domain?

For our analyses, we considered the paired comparison task⁴⁹. Participants in this task decide between two options, each of which is characterized by different feature values. The feature values are associated with a value on an unobserved criterion and participants have to learn which option has the higher criterion. We consider a sequential variant where participants take decisions one at a time with feedback provided on which option had a higher criterion value after each trial.

The strategies people use to make decisions in a paired comparison task are widely contested. While economists have taken a rational perspective that suggests that people weigh the different cues appropriately while making decisions^{47,50}, proponents of ecological rationality have argued that people are incapable of such reasoning due to their cognitive constraints^{51,52}. Instead, they have proposed the view that people rely on simple heuristics, which are short-cut strategies that produce competitive performance despite using only parts of the available information⁵³. Recent work⁴⁹ has shown that people adopt different decision-making heuristics depending on the structure of paired comparison tasks⁵⁴. When participants knew the importance of

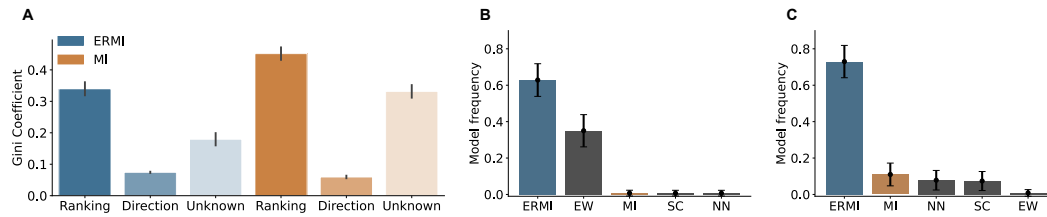


Figure 4. Decision Making: **A** Mean Gini coefficients computed over weights produced ERMI and MI on the tenth trial of the paired comparison tasks sampled from the generative model used in the Binz and colleagues⁴⁹ for three conditions: ranking, direction and unknown. **B** Posterior model frequency of participants' choices from experiment 3b of Binz et al. study⁴⁹, which uses four attributes for each option. **C** Posterior model frequency of participants' choices from experiment 3a of Binz et al. study⁴⁹, which uses two attributes for each option.

attributes to the criterion but not the direction of their correlation with it (ranking condition), they used a one-reason decision strategy. When the direction was known but not the ranking (direction condition), they relied on an equal weighting strategy. Finally, when neither ranking nor direction was known (unknown condition), they used a weighted combination of attributes to guide their choices.

To further examine how data distributional properties influence heuristic choice, we adapted the previously described procedure to generate three sets of problems, each containing approximately 7,000 tasks, reflecting one of three conditions: ranking, direction, and unknown. This was done using three condition-specific prompts specifying: (i) that attributes be rank-ordered by importance to the target; (ii) that attributes correlate positively with the target; or (iii) no additional information to allow free-form generation. We then verified that the generated tasks matched their intended conditions: tasks in the ranking condition showed more rank-ordered feature weights than those in the unknown condition, and tasks in the direction condition had features more strongly (positively) correlated with the target (see Figure S7 in the SI). After that, we constructed paired comparison trials by randomly sampling two options from each dataset and pitting them against each other. We then derived an ERMI model by meta-learning on LLM-generated tasks for each condition.

The resulting ERMI models were first simulated on decision making problems from Binz and colleagues⁴⁹. To examine the strategy being implemented by ERMI, we computed the Gini coefficient over attribute weights produced by ERMI (see Methods for details). Higher values for the Gini coefficient indicate more sparse weights, which corresponds to a one-reason decision making strategy, lower values correspond to equally weighted attributes, and values in between correspond to a weighted-additive strategy. As shown in Figure 4A, we found that ERMI uses the same heuristics that people use in the respective condition. That is, ERMI trained on decision making problems from the ranking condition implements a one-reason decision making strategy ($M_{\text{Gini}} = 0.3399$, $\text{SEM} = 0.0631$), in the direction condition it uses an equal weighting strategy ($M_{\text{Gini}} = 0.0743$, $\text{SEM} = 0.0138$), and in the unknown condition it relies on a weighted combination of attributes ($M_{\text{Gini}} = 0.1794$, $\text{SEM} = 0.0333$). These results are also consistent with the strategies used by meta-learned inference (MI) with a hand-crafted prior for each condition; see Methods for details.

In addition to the simulation study, we evaluated whether ERMI explains human decision making better than competing models by conducting a model comparison on human data from Binz et al.⁴⁹. We considered human data from two experiments, one with options containing two attributes and the other with four, in which participants performed decision-making tasks without receiving any side information (see human studies in SI for details). Compared to other baseline models – namely, single-cue decision making (SC), equal-weighted strategy (EW), feedforward neural network (NN), and MI – ERMI accounts for human responses most frequently in both the two-attribute experiment ($M_{\text{PMF}} = 0.7299$, $\text{SEM} = 0.0888$) and the four-attribute experiment ($M_{\text{PMF}} = 0.6284$, $\text{SEM} = 0.0897$). These results suggest that ERMI converges to the same decision making strategies that people have been shown to use, with the particular strategy shaped by the statistical structure of decision-making problems.

Discussion

In the late 1990s, Gerd Gigerenzer and his colleagues conducted what, in hindsight, stands as one of psychology's most endearingly simple yet profoundly revealing studies. They asked participants whether they recognized the names of various cities or companies—and found that when people chose the company they recognized between two options, their choices reliably predicted which company had higher stock returns. This surprisingly effective strategy, dubbed the recognition heuristic, is a lexicographic decision rule similar to the ones we modeled that stops at the first discriminating cue—in this case, recognition. But how could such a seemingly naive rule succeed in complex contexts like financial forecasting? Gigerenzer proposed that the frequency with which people encounter names in everyday life –on television, in conversations, or in headlines– contains

statistical signals informative for decision making⁵⁵, a hypothesis he substantiated by meticulously counting company name occurrences in major newspapers.

Yet this insight raised a far-reaching question: could one ever scale ecological rationality beyond a single heuristic to explain the full complexity of human learning and decision making? After all, manually tallying newspaper mentions might work for isolated cues, but becomes hopelessly unwieldy when faced with the rich environments people face daily. How could one map the statistical fingerprints of vast environments across countless domains of cognition?

In recent years, an unexpected answer may have emerged: large language models. Indeed, it has been argued, by Alison Gopnik and colleagues⁵⁶, that LLMs, trained on the sprawling archive of human culture, can be seen as “cultural technologies,” artifacts that distill the collective knowledge of societies. Where earlier researchers scraped headlines to estimate how often a name appeared in people’s environments, LLMs now embed billions of such frequencies and co-occurrences, capturing statistical regularities on a scale previously unimaginable. This extraordinary capacity opens, for the first time, an opportunity to massively scale up ecological rationality. We can use LLMs to generate ecologically grounded tasks that reflect the natural statistics of human environments and test whether human learning and decision-making align with ideal inference under such ecological priors.

In the current work, we introduced ecologically rational analysis, a framework that leverages meta-learning and LLMs to do exactly that, i.e. to extend the logic of ecological rationality beyond individual heuristics and into the broader structure of human cognition. In particular, we developed a new class of models—called ERMI—that allowed us to investigate whether human learning and decision making approximate “ideal statistical inference under the structure of natural tasks and environments”⁵⁷. Across 15 experiments spanning three core domains of human cognition, we found that ERMI can account for a substantial amount of variance in human behavior. Not only did ERMI capture key behavioral signatures in each domain, but it also provided superior trial-level prediction of human choices relative to established cognitive models. Taken together, these findings demonstrate that rational adaptation to ecologically valid task statistics is sufficient to account for much of human cognition.

A key strength of ERMI lies in its ability to derive priors and distill them into computational models without extensive hand-engineering. In contrast, traditional rational analysis requires researchers to manually specify the underlying data-generating distribution. For example, in the rational model of function learning, Lucas et al.³⁵ assumed a linearity-based prior but acknowledged uncertainty about its alignment with naturalistic environments, noting that “it is not realistic to directly measure the statistical structure of the environment, that is, what functions are truly more or less common”³⁵. ERMI circumvents this issue by using LLMs to directly generate tasks with ecological statistics. Alternatively, ecological rationality often relies on researchers to manually construct heuristics that are “applicable to specific decision tasks and in particular domains—different tools for different tasks”⁵⁸. By leveraging meta-learning to automatically derive computational models adapted to these ecological statistics, ERMI eliminates the need for hand-designing task-specific heuristics prevalent in ecological rationality framework.

Yet one may ask: Why not directly use an LLM to model human behavior, instead of meta-learning on LLM-generated tasks? We evaluated LLMs as direct behavioral models and found that ERMI consistently outperformed them in explaining human data (see Figure S5 in the SI), highlighting that LLMs do not capture human behavior out-of-the-box. Furthermore, even a strong alignment of LLMs with human behavior would not clarify why they are good models, given that they are trained on vast and opaque datasets that span human conversations, code, and various cultural artifacts that are difficult to analyze. ERMI, on the other hand, leverages LLMs solely as generative sources for ecological tasks used in meta-learning, allowing us to test a specific hypothesis about human cognition. Of course, what features of the environment are required to explain behavior in a particular domain still needs to be determined statistically, which re-introduces a degree of manual analysis that ERMI does not yet fully automate. In that sense, while ERMI reduces the burden of handcrafting heuristics and priors, it does not eliminate the need for scientific judgment in feature selection and interpretation. However, this remaining bottleneck also presents a new opportunity: by systematically varying environmental features across LLM-generated tasks and analyzing their impact on model fit and human behavior, we can begin to reverse-engineer the ecological ingredients that most shape cognition.

Future work should determine which additional components are required to account for human behavior beyond ecological rationality. ERMI offers a flexible foundation for integrating such components. First, we can incorporate participant-specific information into the data generation process, followed by meta-learning on these tailored datasets. This approach would enable personalized ERMI models that capture individual differences, particularly those shaped by environmental and demographic factors unique to each participant. Second, while computational models derived via ERMI currently emphasize adaptation to the environment, they largely ignore the role of cognitive constraints. Incorporating limits on computational complexity—such as attention, working memory, or representational capacity—could help explain additional variance in human behavior, especially in cases where people systematically deviate from ideal inference^{49,59,60}. Notably, such constraints can either be explicitly modeled within the meta-learning process or may already be implicitly embedded in the training data itself, given that these data are generated by humans who are inherently resource-bounded. Third, ERMI could serve as a starting point for fine-tuning on human choice data, following recent approaches^{61–63}. This would allow for principled estimation of residual

variance in behavior not yet captured by ERMI and help identify the cognitive mechanisms needed to close that gap.

Humans excel at learning and decision making in complex and uncertain environments. Our findings suggest that these cognitive abilities emerge largely through attunement to ecological structures. By harnessing ecologically rational analysis, combining psychology and machine learning, we demonstrate how general-purpose models, trained in realistic ecological tasks, can mimic much of human behavior. Looking ahead, we speculate that scaling up this approach to open-ended, embodied environments⁶⁴, which require processing high-dimensional visual information and executing complex control sequences, holds promise for expanding ecologically rational analysis. By leveraging multimodal foundation models to generate personalized, ecologically valid tasks and meta-learning for distilling those priors into adaptive computational models, we can systematically quantify how much of human behavior can be explained as an adaptation to previously encountered task structures and environments. If successful, this would significantly broaden the explanatory power of cognitive models, offering nuanced insights into the ecological roots of human cognition.

Methods

In this section, we first describe how we generate cognitive tasks at a scale that is sufficient to train in-context learning models from scratch and how we verify their ecological validity. Following this, we discuss how to derive in-context learning models via meta-learning and present other domain-specific cognitive models used as baselines. We then describe (a) how we simulate behavior from different models on these experiments, and (b) how we fit and compare these models to human data.

Scalable generation of cognitive tasks from LLMs

Generating cognitive tasks from an LLM entailed a two-stage process. In the first stage, we query an LLM to synthesize the names for input features and targets. For instance, an example input feature in function learning could be CALORIE INTAKE whose corresponding target is WEIGHT. In the second stage, the LLM is queried again but this time to generate numerical values for a given input feature and target pair generated from the first stage. That is, the LLM is tasked to generate different values for [CALORIE INTAKE, WEIGHT], for instance, [2300, 152.0] or [1850, 143.0].

Below, we provide the prompts used in the two stages for the function learning domain; see SI for prompts for other domains. We used the following prompt to synthesize names for input features and targets for function learning:

Synthesize input feature name and its target

I am a psychologist who wants to run a function learning experiment. In a function learning experiment, a real-world feature is mapped to its corresponding target, with both feature and target taking on continuous values.

Please generate names for features and its corresponding target for **250** different function learning experiments:

– feature name, target name

Next, we prompted the LLM to generate values for a function learning task generated from the first stage:

Generate values for a given function learning task

I am a psychologist who wants to run a function learning experiment. For a function learning experiment, I need a list of features with their corresponding target. The feature in this case is **calorie intake**. The features take on only numerical values and must be continuous. The target, **weight**, should be predictable from the feature values and must also take on continuous values.

Please generate a list of **20** feature-target pairs sequentially using the following template for each row:

– feature value, target value

We generated a dataset containing around 10000 different function learning tasks with each task consisting of 20 data points from CLAUDE-v2⁶⁵. The temperature parameter was set to one to induce diversity, and all other parameters were set to their default values. We chose CLAUDE as it can process up to 100,000 tokens, is instruction-tuned, cost-effective, and performed well out of the box on most of our preliminary analyses; see SI for information about the other LLMs we considered.

To use and analyze the generated cognitive tasks, we parse all necessary quantities from the output text from the LLMs using regular expressions and stored them in numerical format in comma-separated-value (csv) files. These stored csv files

are the datasets we use for further analysis. We expand on the parsing expressions, data-processing steps, and also provide a qualitative analysis of synthesized input feature and target names (see Figures S1, S3, S4, and S6) in the SI.

Verifying the ecological validity of LLM-generated cognitive tasks

To test the ecological validity of the generated cognitive tasks, we resort to two approaches. We either compare certain key statistics between LLM-generated tasks and a real world baseline, whenever we have access to a reasonable dataset, or compare it to real world statistics expected or predicted by prior work. We will discuss these tests for each domain individually below.

Function learning.

We compared the data distributional properties of the LLM-generated function learning problems with 60 real-world regression tasks curated by Lichtenberg and colleagues²⁵. We downsampled all tasks in the dataset to a single input dimension by applying univariate feature selection using the F-statistic for regression, as implemented in SCIKIT-LEARN⁶⁶, and included only tasks without missing values and with at least one valid input dimension in our analysis. Note that each dataset was split into separate tasks of 25 datapoints each, yielding a collection of regression problems with fixed size for analysis.

For both real and LLM-generated function learning tasks, we estimated the relative frequency of different function classes within the dataset. We did this by first fitting models of different function classes to each LLM-generated task and then, assigning the function class with the best fitting model to the given task.

Specifically, we considered models from four well-studied function families, namely, linear, exponential, quadratic, and sinusoidal, from the literature²⁶; see SI for their exact model instantiations. The parameters of these models, ϕ , were fit to data from the task to minimize the sum of squared errors (SSE) using the curve fit function from the SCIPY optimization library⁶⁷. We then computed the Bayesian Information Criterion (BIC) for the fitted models from each function class, compared them against each other, and assigned the label of the function class that won the model comparison to a given task. Assuming $\hat{y}(\phi)$ and y correspond to predicted target from the fitted model with parameters ϕ and true target, respectively, BIC computation entailed the following steps:

$$\begin{aligned} \text{SSE} &= \min_{\phi} \sum_{i=1}^N (y_i - \hat{y}_i(\phi))^2 \\ \text{BIC} &= N \cdot \ln(\text{SSE}) + |\phi| \cdot \ln(N) \end{aligned} \quad (1)$$

where $|\phi|$ is number of parameters in a given model parameters, and N is number of data points per task. This SSE-based approximation of the BIC assumes that model errors are Gaussian with constant variance, under which the negative log-likelihood is proportional to the sum of squared errors.

We obtained the proportion of different function classes by computing a histogram over the assigned class labels for all tasks in a given dataset. Furthermore, we assessed whether the fitted slope term of the linear model were predominantly positive and whether the fitted offset term, from the same model, was close to zero.

Category learning. We compared the data distributional properties of the LLM-generated category learning tasks with a real-world classification benchmark⁶⁸. For this, we used the OpenML-CC18 benchmarking suite, a curated collection of real-world classification tasks¹⁴. We downsampled all tasks in the OpenML-CC18 benchmark to four feature dimensions by applying univariate feature selection using the ANOVA F-test implemented in SCIKIT-LEARN⁶⁶ and included only binary classification tasks without any missing features in our analysis – amounting to 28 tasks.

We analyzed these collections of tasks in terms of their learning curves, input feature correlations, sparsity of predictive features, and linearity of the category structure. We obtained the learning curves by fitting a logistic regression model on a trial-by-trial fashion. For input correlations, we computed Pearson’s correlation coefficient between every pair of features in the task. To get an estimate for task sparsity, we fitted a logistic regression model on the full data for each task and analyzed the sparsity of the resulting regression weights $\mathbf{w} \in \mathbb{R}^d$ using the Gini coefficient G :

$$G(\mathbf{w}) = \frac{\sum_{i=1}^d \sum_{j=1}^d |\mathbf{w}_i - \mathbf{w}_j|}{2d \sum_{i=1}^d \mathbf{w}_i} \quad (2)$$

For determining the linearity of the category structure, we fitted a logistic regression model and a logistic regression with second-order polynomial features on the full data $\mathcal{D}$ from each task. We then computed the BIC for both models and used them

to approximate the posterior probability that the linear model offers a better explanation of the data (assuming a uniform prior over models), see Equation 3.

$$p(M = \text{linear} | \mathcal{D}) \approx \frac{\exp(-0.5 \cdot \text{BIC}_{\text{linear}})}{\sum_{m \in \{\text{linear}, \text{polynomial}\}} \exp(-0.5 \cdot \text{BIC}_m)} \quad (3)$$

Decision making. We examined the distribution of input feature correlation, sparsity in predictive features, rank ordering of feature importance, and directionality of the features for the three LLM-generated decision making datasets belonging to ranking, direction, and unknown condition; see SI for details about their generation. For baseline, we considered the LLM-generated dataset in the unknown condition, as it allows contrasting dataset from the rank and direction condition with one that lacks explicit manipulation. See SI for data-distributional properties of LLM-generated decision making tasks.

For measuring correlation between input features, we compute pair-wise Pearson’s correlation coefficient, following the same procedure we used in the domain of category learning; see Figure S7 (first column) in the SI for visualization.

To measure sparsity of task features, we followed the same procedure as in the category learning task but instead of a logistic regression model, we fitted a linear regression model that predicts the continuous valued targets from the task features; see Figure S7 (second column) in the SI for visualization.

For examining the rank ordering of feature importance, we fit a linear regression model, predicting the target from the input features. We then identified the feature with the highest absolute regression coefficient for each task and performed histogram over these positions to assess how often each feature was most predictive. If the intended manipulation was successful, we expect that the first feature should most frequently have the largest coefficient, followed by the second, and so on, reflecting a consistent ordering of feature relevance; see Figure S7 (third column) in the SI for visualization.

We assessed the directionality of each feature by examining the sign of the fitted regression coefficients from linear models as described above. If all coefficients are positive, it suggests that the intended manipulation was successful; see Figure S7 (fourth column) in the SI for visualization.

Ecologically Rational Meta-learned Inference

Having generated and tested the ecological validity of LLM-generated cognitive tasks, we then trained transformer-based models on those tasks to derive explicit in-context learning models adapted to the ecological task distribution. For this, we let a transformer-based model¹⁵ auto-regressively predict a target, y_t which can either be a discrete category or a continuous response, for a given input, x_t , conditioned on all preceding input-target pairs, $(x_{1:t-1}, y_{1:t-1})$. After the model predicts targets for all inputs in the sequence, the parameters of model, θ , is updated based on the following objective:

$$\ell = \sum_t -\log p_{\theta}(y_t | x_{1:t}, y_{1:t-1}) \quad (4)$$

where p_{θ} defines the output probabilities produced by the model.

The model is then trained until convergence, such that post convergence it can perform in-context learning. That is, the model can learn to predict the correct target for a new input based on previously seen input-target pairs – provided in context. Critically, in-context learning is implemented by the model purely via its internal activations, without any additional weight updates after training. Previous work has demonstrated that this form of explicit in-context learning algorithms approximates the Bayes-optimal learning algorithm on the distribution of tasks $p(x_{1:T}, y_{1:T})$ encountered during training¹². This key result enables us to make links between in-context learning displayed by our models and rational analysis⁶⁹.

The base neural network in our in-context learning models was the transformer-based decoder architecture¹⁵ with a causal attention mask, as done previously^{10,18,70}. The network settings were chosen based on a hyper-parameter search and were different for each domain (see SI for details), but all models irrespective of domain used positional encoding based on sine and cosine functions of different frequencies¹⁵. For training, a batch of tasks is sampled from $p(x_{1:T}, y_{1:T})$ in each episode and the model predicts the target for the given input conditioned on all preceding input-target pairs. After which, model parameters are updated based on the objective mentioned in Equation 4 using a schedule free optimizer⁷¹ with the learning rate set to $3e-4$. We provide additional details about the model architecture and training procedure in the SI.

Baseline models

We chose several domain-specific cognitive models and compared them with ERMI; see SI for full details. For function learning, we compared against a meta-learned inference (MI) model trained on functions drawn from a hand-crafted prior distribution over kernels. Following Lucas et al.³⁵, the prior probabilities for positive linear, negative linear, quadratic, and radial basis kernels were set proportional to 8, 1, 0.1, and 0.01, respectively.

We considered six models for the domain of category learning, namely, the rational model of categorization (RMC⁴³); a meta-learned inference (MI) model trained on synthetically generated problems with linear decision boundary; a meta-learned

inference model trained on synthetically generated tasks with non-linear decision boundary (PFN⁷²); the generalized context model (GCM⁴⁵), a prototype model (PM⁴⁴), and a rule-based learning model (Rule³⁴).

Four models were considered for the decision making task. First, a meta-learned inference model trained on synthetic decision making problems sampled from the true generative model used in the experiment (MI⁴⁹). Second, a single-cue decision maker (SC⁴⁹). Third, equal weighting decision maker (EW⁴⁹). Fourth, a feedforward neural network (NN⁴⁹).

Model simulations

In this section, we provide details of how model simulations were performed for the different experiments reported in this study.

Function learning

Learning difficulty and speed. To generate the learning difficulty curves shown in Figure 2C-D, we first sampled functions, y , from linear positive, linear negative, exponential, quadratic, and periodic families, for different values of input, x , ranging between 0 and 1. For the linear functions, we used the functional form $y = mx + c$, where we sampled slope and intercept terms from uniform distribution between -1 and 1. For the exponential functions, we used $y = a * e^{bx+c} + d$, where the terms were all sampled from uniform distribution between -1 and 1. Quadratic functions used the following parameterization: $y = w^2 + c$, with values for parameters sampled from uniform distribution between -1 and 1. We used the functional form: $y = w * \sin(2\pi x - \phi) + c$ for periodic function with amplitude, frequency, phase and offset sampled from uniform distribution between -1 and 1. All values were chosen such that final values are in the range between -1 to 1. We obtained the targets for each input value auto-regressively, conditioned on previous inputs and targets. We run this simulations 100,000 times for both ERMI and MI, and report the mean over trials for both models.

Interpolation and Extrapolation. We considered the same exact linear functions with fixed offset as used in the original Kwantes et al.²⁷ study. We only additionally normalized the input and target to be between -1 and 1, such that it matches the range of inputs taken by ERMI during training. We extracted ERMI's and MI's predictions auto-regressively, conditioning on previously observed input-target values.

Category learning

Learning difficulty. To run simulations of the Shepard study³⁷ on ERMI and MI, the geometric inputs used in the original study were converted into binary coded vectors taking values along the three input feature dimensions. The value assignment for a input feature was randomized in every run, the order of presentation of the input was also randomized, and the number of presentations of a input per block was matched to the original study.

Learning strategy. The 616 choices made by ERMI and MI were divided into 11 blocks of 56 trials each. The choices were obtained from the model by simulating them on a numerically abstracted version of the task, similar to the learning difficulty mentioned above. The simulations were run for a total of 50 runs using the softmax temperature term fitted to participants in the Devraj et al. 2021⁴¹ study. We then fit prototype-model (PM) and exemplar-based model (GCM) onto the choices of humans and models to see if they are better explained by prototype or exemplar-based strategy. To fit their parameters, we minimize the sum of squared errors (SSE) between observed and predicted probabilities for each participant for a given block following the original study's approach:

$$SSE = \sum_{t=1}^{14} (p(y_t = 1|x_t) - \hat{p}_{1,x_t})^2 \quad (5)$$

where $p(y = 1|x_t)$ is the predicted probability from the model – either GCM or PM – that input x_t belongs to category 1 based on an entire trial segment (56 trials) of data, and $\hat{p}_{1,x_t}$ is the proportion of trials in the trial segment (out of those in which input x_t was seen) in which the participant or model categorized input x_t to category 1. We used SciPy's Sequential Least Squares Programming (SLSQP) method to obtain the best fitting parameter for the two models as in the original study⁴¹. We then compared the SSE computed using the best-fitting parameters between the two models as shown in Figure 3.

Generalization. We simulated ERMI and MI on the Johansen et al. study³⁹ for inverse temperature values, from zero to one in steps of 0.1, for a total of 544 runs. The models interacted with each of the nine training inputs 32 times, with the ordering of the inputs shuffled between runs. Predictions for the transfer inputs were derived by concatenating them – one at a time – at the end of 32 training blocks in every run. By doing so, we were able to derive the model's prediction for each unseen input around 77 times. In Figure 3, we reported average choice probabilities for the models using the inverse temperature value that minimized the pair-wise Euclidean distance between the human and model's choice probabilities.

Decision making

We evaluated ERMI and MI model on paired comparison tasks following the same generative model used as the original study⁴⁹. ERMI and MI took values for the four attributes for both options along with the correct target from the previous trial as input at

each step. They then predicted one of the two options on the current step. The simulation was performed for the same number of trials and blocks as in the original study.

Model fitting and comparison

Parameters for models considered in this work were fit to the data using maximum likelihood estimation. The exact model parameters fitted for each model and their implementation details are discussed in the SI.

After fitting the models, we performed a Bayesian model comparison, with goodness-of-fit to human choices measured based on posterior model frequency⁷³. The posterior model frequency measures how often a given model offers the best explanation in the population. We computed it using a Python implementation of the Variational Bayesian Analysis (VBA) toolbox⁷⁴; see SI for additional details.

Data, software, and code

Data, code, and analysis scripts are available at <https://github.com/akjagadish/meta-learning-ecological-priors-from-llms/>

Acknowledgements

We thank the members of the Institute for Human-Centered Artificial Intelligence (HCAI) for their comments, discussions, and support throughout this work. Special thanks to Milena Rmus for helping out with Figure 1. We also specifically thank Devraj et al.⁴¹, Nosofsky et al.⁴⁰, Binz et al.⁷⁵, Little et al.²⁹, and Badham et al.⁴² for making their data publicly available. This work was supported by the Max Planck Society, Helmholtz Center, Volkswagen Foundation, Princeton University, and Deutsche Forschungsgemeinschaft (DFG) under the German Excellence Strategy - EXC 2064 / 1 - 390727645.

Author contributions statement

A.K.J., M.B., and E.S. conceived the study and developed the methodology and theoretical framework; J.C. and M.T. contributed to refining the theoretical framework; A.K.J. designed and conducted the experiments with contribution from M.B.; A.K.J. collected and preprocessed the data and generated the figures; A.K.J. wrote the original draft of the manuscript; A.K.J., J.C., M.T., E.S., and M.B. reviewed and edited the manuscript; E.S. acquired funding.

Supplementary Information

Function Learning

LLM-generated tasks

The exact prompts and data generation pipeline for function learning are discussed in the Methods section of the main text.

Parsing synthesized task features and labels: We queried CLAUDE-V2 to generate feature names in the format: FEATURE DIMENSION 1, FEATURE DIMENSION 2, ...; see Methods in the main text for exact prompts. To extract these, we used a series of regex patterns, such as

([A-Za-z&]+), ([A-Za-z&]+) and its higher-arity extensions, designed to capture up to five comma-separated alphanumeric feature names (including symbols like “&”). These patterns allowed us to reliably extract structured feature descriptions across tasks. The parsed feature names were stored in a dataframe for subsequent task construction and evaluation.

Qualitative analysis of synthesized task features and labels: We show the counts for the top-50 most frequently occurring names for (a) inputs and (b) targets in Figure S1. We found that LLM tends to produce input-target pairs belonging to everyday topics such as education (practice time versus skill), health (calories burned versus weight change), agriculture (rainfall versus crop yield), etc.

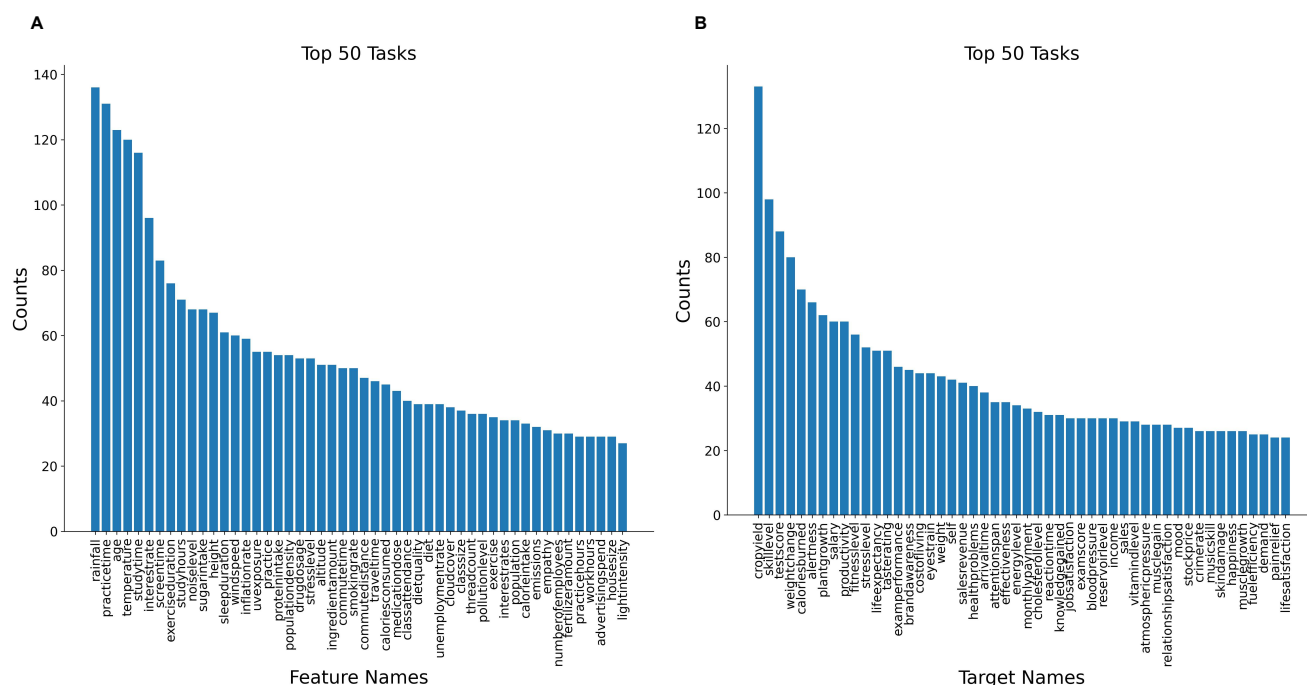


Figure S1. Frequency of input and target labels in CLAUDE-v2 synthesized function learning tasks: Counts for the top-50 most frequently occurring (a) input and (b) target labels computed over 9991 LLM-generated function learning tasks. These distributions confirm that the LLM-generated tasks capture real-world functional relationships.

Parsing generated task data points: CLAUDE-v2 was prompted to generate datapoints in the format: - FEATURE VALUE 1, FEATURE VALUE 2, ..., FEATURE VALUE N, TARGET VALUE; see Methods in the main text for exact prompts. To extract numerical values from these responses, we constructed regex expressions of the form $([\backslash d.]^+)$, repeated for each feature dimension, followed by $([\backslash d.]^+)$ for the target value. This pattern reliably captured sequences of decimal numbers across varying dimensionalities. The extracted values were stored in a dataframe, serving as a structured dataset for training and evaluating meta-learned function approximators.

Data processing: We filter out all tasks containing more than 20 data points to ensure consistent task lengths and evaluation settings. We randomly shuffled the trial order within each task. We resampled the trials with replacement to match the target task duration, enabling evaluation on experiments with longer trial horizons without performance degradation. All

feature dimensions were independently normalized to lie within $[-\text{scale}, \text{scale}]$ using a Min-Max normalization scheme, where $\text{scale} \in [0.1, 0.5]$ was fixed or randomly sampled. LLM-generated tasks can sometimes be of varying lengths, and in the case that the task length was shorter, they were padded with zeros to match the longest task in the batch. The maximum steps or number of trials for the experiment we considered was 25 trials. The batch size was fixed to 64 unless otherwise specified.

Models fit to LLM-generated data: We considered models from four well-studied function families, namely, linear, exponential, quadratic, and sinusoidal, as mentioned in the Methods. For the linear function, we chose the instantiation $y = a * x + b$, with initial parameters set to 1 and 0 for the slope and offset terms, respectively. For quadratic, we chose $y = a * x^2 + c$, with initial parameters for slope and offset set to 1 and 0 respectively. We chose $y = a * \exp(b * x) + d$ for the exponential family, with initial parameters for a set to the mean difference between the maximum and minimum of the target values, b set to 1, and offset term set to the minimum of the target values. We chose $y = a * \sin(b * x) + d$ for the sinusoidal family with initial parameters a set to the mean difference between the maximum and minimum of the target values, b set to $2 * \pi$, and offset term set to the mean of the target values. We fit the parameters of these models to LLM-generated functions using the curve fit function from the SCIPY optimization library⁶⁷.

Human studies

Kwantes and Neal 2006²⁷. In this study, 14 participants had to learn to predict values along the y-axis for different values on the x-axis, with samples drawn from a linear function $y = 2.2x + 30$. Before test phase, they were trained on 20 samples on the x-axis drawn such that their values on the y-axis were always in the range between 30 and 70. The samples were fixed but their order used for training was randomized per participant and session. In each trial, participants made their prediction by entering their estimate as numbers and locking in their answer by clicking on a button labeled “submit your answer”. After locking in, feedback was provided regarding their performance (in terms of accuracy score out of 100). Once training was complete, participants were shown 45 samples in the range from 0 to 100 and asked to enter their estimates. The presentation of the 45 samples were blocked into three sets of 15: low (0-30), medium (30-70), and high (70-100) range. The order in which the blocks were presented and the order of samples within them was randomized for each participant.

Little et al. 2024²⁹. This study was conducted on 177 participants. The particular experiment we consider, called function estimate test, was included as part of larger paper-based questionnaire. In this experiment, participants were presented 24 scatter plots, each depicting data from a different fictional scientific experiment, on a piece of paper, with two 7.5 cm by 7.5 cm graphs in each page with 4 cm gap between them. They were then instructed to draw the true underlying causal function for the data points in the graph. The data points could be presented in either large (zoomed in version), where the data points covered the entire figure, or small (zoomed out version), where it covered 40 percent of the total area, scale. The relative position of the points in the small- and large-scale sets was kept identical. Three functions were used to generate data for the scatter plots, namely, linear, quadratic, or cubic polynomial functions. A small amount of Gaussian noise was added as jitter in all graphs. The data points and the drawn functions used for model fitting were extracted from scans of the physical document using a software program called Data Thief⁷⁶. After extraction, the data was down-sampled to include 40 evenly spaced data points in the range of the x-axis and with all points scaled to be between -1 and 1. We used the data from the following [GitHub repository](#).

Hand-crafted tasks

Functional priors from rational model of function learning³⁵ used for training ML model: We generate 10,000 synthetic regression tasks for function learning using a mixture of kernels adapted from the study by Lucas et al.³⁵. Each task involved a one-dimensional input sampled from a uniform grid of 20 points in the interval $[-1, 1]$. The target output was computed by sampling a kernel type from a hand-crafted prior: favoring positive linear (probability 0.8), followed by negative linear (0.1), quadratic (0.01), and radial basis (0.001) functions and applying the corresponding transformation to the input. Parameters for each kernel (e.g. weights, intercepts, distances) were drawn from a gamma distribution with shape 1.001 and scale 1.0. A small amount of Gaussian noise was added to the target. All inputs and targets were dynamically scaled to lie in $[-\text{scale}, \text{scale}]$, where the scale is sampled from a uniform distribution in the range $[0.1, 0.5]$ in each training batch.

Model architecture, and training

Each trial in a function learning task consisted of an input vector concatenated with the previous target value, and these were embedded into a 64-dimensional space. Positional encoding was applied using sine and cosine functions of varying frequencies, following Vaswani et al.¹⁵. A causal attention mask ensured that predictions at each time step were conditioned only on past inputs and targets. These masked sequences were processed using a Transformer decoder composed of six layers, with 64-dimensional embeddings, eight attention heads, and 256 hidden units in the feedforward layers. The decoder outputs were passed through two independent linear projections to produce the mean and standard deviation of a normal distribution. The negative log-likelihood (NLL) was computed over all targets in a given batch, and minimizing it served as a loss function for

training the network parameters. The model parameters were updated using the SCHEDULEFREE optimizer⁷¹ with a baseline learning rate of 3×10^{-4} . Each model was trained for 250,000 episodes, with periodic evaluation on held-out tasks to monitor generalization performance.

Model fitting and comparison

We did not fit any model parameters to human data in both ERMI and MI. We computed the response from these models by querying it on new inputs while being conditioned on the input-target pairs participant observed before drawing the functions. For model comparison, we report the mean-squared error between the participant's actual response, sampled from the functions they drew through the data points displayed to them, and model predicted target for the same input.

Additional results

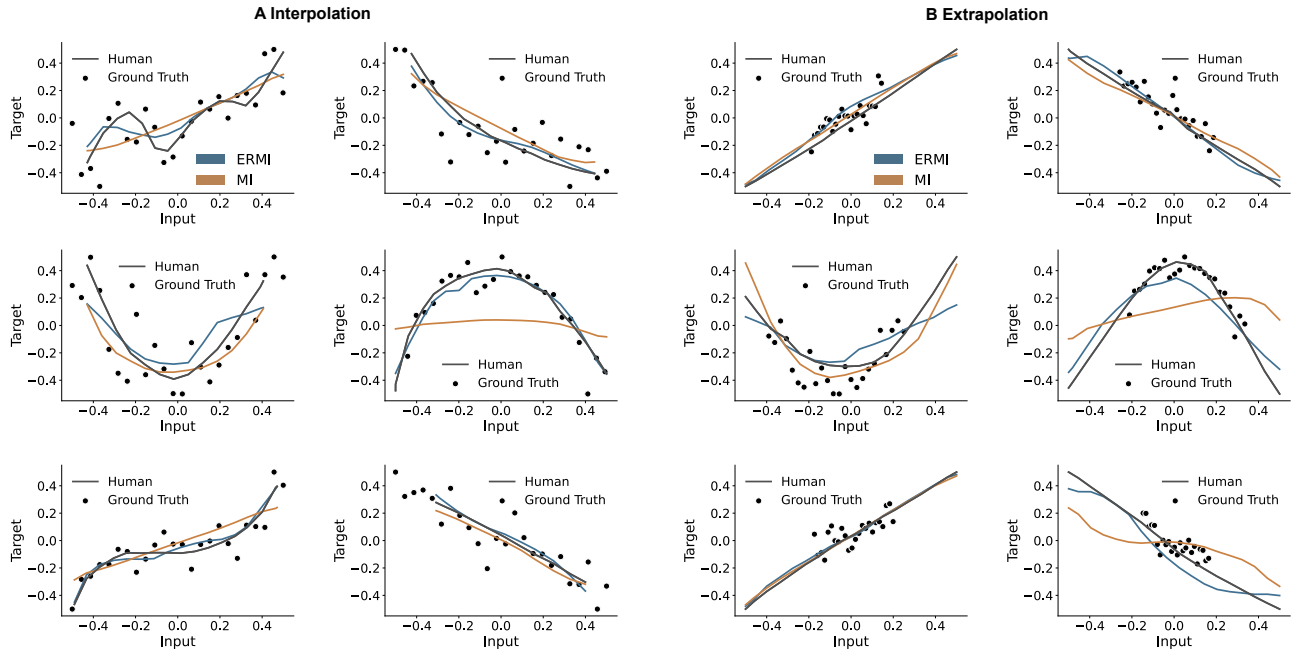


Figure S2. Predictions derived from ERMI and MI for function families used in the Little et al. study²⁹ for both interpolation (zoomed in; A) and extrapolation (zoomed out; B) condition. The function families considered were linear (top row), quadratic (middle row) and cubic polynomial (bottom row); see Human studies section in Methods for details.

Category Learning

LLM-generated tasks

Prompts: We used the following prompt to synthesize feature names and category labels for the category learning task.

Synthesize feature names and category labels

I am a psychologist who wants to run a category learning experiment. In a category learning experiment, there are many different three-dimensional stimuli, each of which belongs to one of two possible real-world categories.

Please generate names for three stimulus feature dimensions and two corresponding categories for 250 different category learning experiments:

In the second stage, we prompted the LLM to generate data points for the synthesized features and the category label. Below is the prompt, for a category learning where the synthesized input features were sodium, fat, and protein, and categories are healthy or unhealthy:

Generate category learning tasks

I am a psychologist who wants to run a category learning experiment. For a category learning experiment, I need a list of stimuli and their category labels. Each stimulus is characterized by three distinct features: **sodium**, **fat**, and **protein**. These features can take only numerical values. The category label can take the values **healthy** or **unhealthy** and should be predictable from the feature values of the stimulus.

Please generate a list of 100 stimuli with their feature values and their corresponding category labels using the following template for each row:

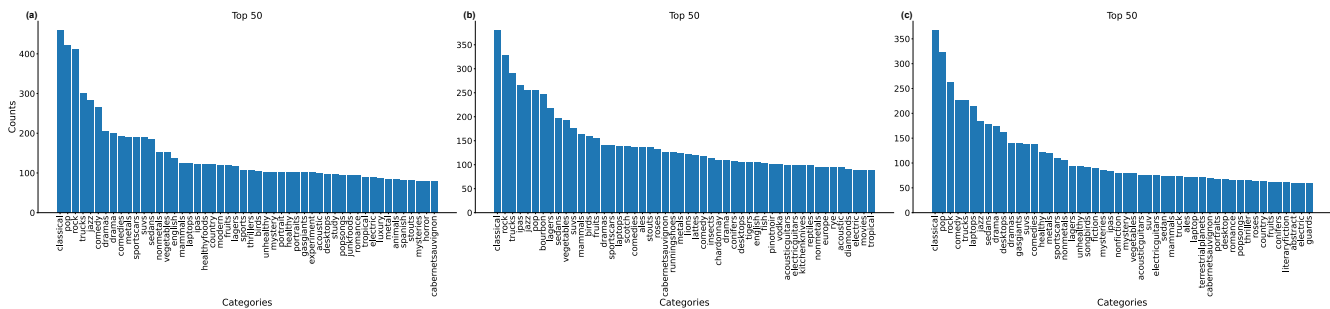
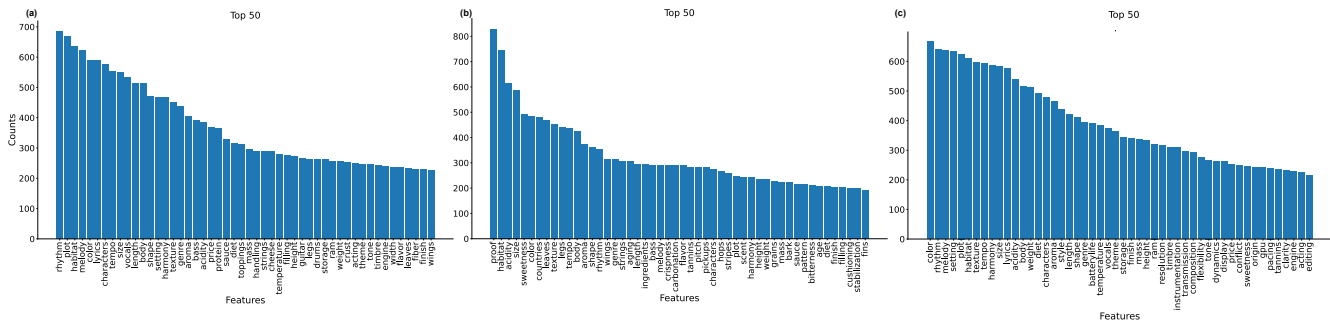
– feature value 1, feature value 2, feature value 3,
category label

Parsing synthesized task features and labels: We prompted CLAUDE-V2 to generate task features and labels in the format: FEATURE DIMENSION 1, FEATURE DIMENSION 2, ..., FEATURE DIMENSION N, CATEGORY LABEL 1, CATEGORY LABEL 2. We extracted relevant entries using the regex pattern `\d+ . ( . + ? ) \n`, which captures text following a numbered bullet point up to the first newline. The resulting string was split at the commas to separate feature names from category labels. All parsed information was stored in a dataframe for downstream use.

Qualitative analysis of synthesized task features and labels: We show the counts for the top-50 most frequently occurring input feature names in Figure S3 and category names in Figure S4 for the 23421, 20690, and 13693 category learning tasks generated with three (a), four (b) and six-dimensional input features, respectively. When it comes to input feature names, we found that the LLM tends to produce features belonging to topics such as musicality (like rhythm, melody, lyrics, tempo, vocals), food (like aroma, texture, crust, diet, protein), etc. With regard to category names, there were also many related to music (for example, classical, pop, jazz, rock), but also vehicles (like trucks, SUVs, sedans), technology (laptops, desktops, iPads), etc.

Parsing generated task data points: To generate data points for each task, we queried CLAUDE-V2 using the format: - FEATURE VALUE 1, FEATURE VALUE 2, ..., FEATURE VALUE N, CATEGORY LABEL. The model reliably followed this format. To parse the resulting output, we used a suite of regex patterns designed to handle diverse data formats, including numeric values (with or without decimals), alphanumeric labels, hyphens, and various delimiters. Table S1 lists all the regex patterns employed. These enabled us to successfully parse 95% of the generated tasks. The parsed values were stored in a dataframe, forming an offline task repository to train the ecologically rational meta-learned inference model.

Data pre-processing: We filter out all tasks with more than two unique category labels and then binarize the category labels, which are originally strings, to make them consistent across tasks. The assignment of category labels, that is either '0' or '1', within a category learning task was randomized during batch creation. This ensures that there can be no unintended correlations between the inputs seen during training and the labels (across all training data each input vector is assigned half of the time to label '0' and half of the time to label '1'). We also normalized each feature independently using a min-max normalization scheme such that values taken by any feature lie always between zero and one. Both the task features and data points were shuffled while generating tasks. Note that the tasks generated by LLMs are typically of different lengths. Whenever the



sampled tasks are of variable lengths, they are padded with zeros to match the length of the longest task sample within the batch. We additionally also sampled LLM-generated data points with replacement to match the length of the experimental task used in the Devraj et al. 2021⁴¹ and Johansen et al. 2002³⁹ studies. We resorted to this strategy as the LLM-generated tasks had a maximum of about 200 data points per task and by resampling, we can evaluate the model on experiments with larger horizons without any drop in performance. The batch size was set to 64 for three- and four-dimensional inputs and to 32 for six-dimensional inputs and it operated under a maximum steps regime of 400, 300, and 650 for three, four, and six-dimensional tasks respectively.

Human studies

Nosofsky et al. 1994⁴⁰. In their replication of the Shepard et al. 1961³⁷ study, Nosofsky and colleagues conducted the study on 120 participants. The authors used geometric inputs that varied in shape (squares or triangles), interior line type (solid or dotted), and size (large or small). In total, 40 participants performed each of the six category structures, considered in Shepard et al. 1961³⁷. The participants were informed that the rules for each problem were independent. Following the same methodology as Shepard et al., the learning process involved classifying inputs into two categories and receiving feedback. This process was repeated over several blocks (containing up to 16 trials) with randomized input order in each block. Learning in the task was measured until participants achieved a no-error streak in four consecutive sub-blocks of eight trials or reached a maximum of 400 trials. In tasks belonging to TYPE 1, inputs were assigned to a category depending on the values they take along one of the three dimensions, whereas in TYPE 2 tasks, inputs were assigned to a category by applying the exclusive-or rule along two relevant dimensions. Category assignment in tasks belonging to TYPE 3, TYPE 4, and TYPE 5 used a unidimensional rule-plus-exception structure with some inputs grouped in the central region and some in the periphery. Lastly, TYPE 6 tasks required considering feature values along all dimensions, and they require the memorization of every item and its associated category to solve them correctly. For the illustration of category structures for the six types, please refer to Figure 1 in Nosofsky et al. study⁴⁰.

Table S1. Regular expression patterns used for parsing the data points generated for category learning tasks by CLAUDE-V2

INDEX	REGULAR EXPRESSION
1	$([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash w]^+)$
2	$([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w]^+)$
3	$([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+)$
4	$([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+)$
5	$([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+)$
6	$(?:.*?:)?([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$
7	$([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$
8	$r'^{\wedge}(\backslash d+):([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash w]^+)$
9	$r'^{\wedge}(\backslash d+):([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w]^+)$
10	$r'^{\wedge}(\backslash d+):([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+), ([\backslash -\backslash w\backslash d, .]^+)$
11	$r'^{\wedge}(\backslash d+):([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+)$
12	$r'^{\wedge}(\backslash d+):([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+)$
13	$r'^{\wedge}(\backslash d+):(?:.*?:)?([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$
14	$r'^{\wedge}(\backslash d+):([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$
15	$\wedge(\backslash d+):([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash d.]^+), ([\backslash w]^+)$
16	$\wedge(\backslash d+):([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w\backslash -]^+), ([\backslash w]^+)$
17	$(\backslash d+):([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+), ([\backslash ^,]^+)$
18	$(\backslash d+):([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+), ([\backslash ^, \backslash n]^+)$
19	$(\backslash d+):(?:.*?:)?([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$
20	$(\backslash d+):([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+), ([\backslash ^, -]^+)$

Badham et al. 2017⁴². In this study, the authors partially replicated the original Shepard et al. 1961³⁷ study by running it on 96 adults aged between 18 to 87 years. As inputs, they used eight geometric shapes varying in size (large or small), shape (square or triangle), and color (black or white) shown on a mid-gray background. The order of inputs and their category assignment were randomized. Unlike the original study, the authors only considered the first four types of category structures but with the advantage that all participants performed all four types. Participants performed each task type for a total of six blocks with each block containing 16 trials (resulting in a total of 96 trials) or until they reached a criterion of perfect performance in two consecutive blocks.

Smith et al. 1998³⁸. The study was run on 32 participants, where each participant was presented 14 different six-dimensional inputs, with each input mapping to a six-letter nonsensical word such as gafuzi, kafitdo, nivety, wysero, etc (see Appendix A of³⁸ for all words). For modeling, we represented each input as a six-digit binary string, where each digit and position corresponds to a specific letter. For instance, assuming the input ‘gafuzi’ corresponds to the binary code ‘000000’, ‘gyfuzi’ corresponds to ‘010000’, and so on. The inputs were assigned to categories such that input ‘000000’ corresponds to category 1 and input ‘11111’ corresponds to category 2. In this work, we only considered data from the non-linearly separable (NLS) category structure from Experiment 2. In this category structure, each category consisted of six inputs with five features in common with the prototype, and one input with five features in common with the opposing prototype. For instance, if category 1 contained seven inputs as follows: [000000, 100000, 010000, 001000, 000010, 000001, 111101]. The remaining seven inputs belonged to category 2 [111111, 011111, 101111, 110111, 111011, 111110, 000100]. Participants had to categorize a input into one of these two categories and had unlimited time to make their choices. After making their choice, they were told if it was a correct decision or not. Participants completed a total of 560 trials over 10 blocks of 56 trials each. In each block, participants saw each input four times.

Devraj et al. 2021⁴¹. Devraj and colleagues replicated a study of Smith et al. 1998³⁸ and collected data from 60 participants. Participants were recruited from the 18-23 age range and English-speaking population using Prolific. Their study involved 11 blocks and had 616 trials in total. We used the data from the following [GitHub repository](#).

Johansen et al. 2002³⁹. Johansen and colleagues conducted their categorization study on 130 participants in which they presented four-dimensional inputs with each dimension taking binary values. Each of the inputs was a computer-generated

drawing of a rocket that varied along four binary-valued dimensions: The shape of the wing (triangular or rectangular), tail (jagged or boxed), nose (staircase or half-circle), and porthole (circular or star)³⁹. The authors used the same category structure as those used in previous studies^{77,78}. This category structure is ill-defined in that no single feature along a dimension can be used to perfectly classify inputs. Instead, the categories have a family resemblance structure in that category 1 inputs tend to have a value of 0 along each dimension, and category 2 inputs tend to have a value of 1 along each dimension. More concretely, they assigned their five inputs [0001, 0101, 0100, 0010, 1000] to category 1 and the remaining four inputs [0011, 1001, 1110, 1111] to category 2. The inputs were presented serially with their order randomized within each block. Participants had unlimited time to make their choice and were informed whether or not it was a correct choice after each choice. Participants completed a total of 288 training trials, or 32 blocks of 9 trials each, in which they saw each input once. In addition to the training block, participants had to perform a transfer block after 2, 4, 8, 16, 24, and 32 blocks of training. In a transfer block, the eight training inputs along with eight other unseen transfer inputs were shown without corrective feedback. The encoding for transfer inputs, labeled T1 to T7, were (in order): [1011, 1010, 0111, 1101, 1100, 0110, 0000]. It is the category assigned in the transfer block, which is of major interest in this work.

Hand-crafted tasks

Bayesian logistic regression prior used for training MI model: We generated 10,000 synthetic binary classification tasks with a linear decision boundary using a Bayesian logistic regression model. To do this, we sample the input features from a normal distribution with zero mean and unit variance for a given number of data points and input dimensions. We then applied a linear transformation, followed by a sigmoid function, and rounded the result to determine the binary class for the given input. The parameters of the linear transformation are sampled from a normal distribution with zero mean and unit variance. The maximum number of data points within a task was set to 400, 650, or 300 for category learning tasks with three-, four-, and six-dimensional inputs, respectively. These values were chosen according to the length of the experiments on which these models were evaluated.

Bayesian neural network prior used for training prior-fitted network (PFN) model: We generated 10,000 synthetic binary classification tasks using a version of the Bayesian neural network (BNN) developed by Müller et al.⁴⁶. We used normally-distributed i.i.d. input features for a given number of data points and input dimensions. We then passed the input through a BNN with two layers with tanh non-linearity and hidden dimensionality of 64. The network weights and biases were sampled from a normal distribution with a mean of zero and a standard deviation of 0.1 and subjected to an additional sparsity constraint (i.e., 20 percent of randomly chosen network weights and biases set to zero). The maximum number of data points was once again set to 400, 650, or 300 for category learning tasks with three-, four-, and six-dimensional inputs, respectively. The model output is passed through a sigmoid function to generate probability estimates, which are then rounded to determine the class for the given input.

Model architecture, and training

The task features, which contain values for the different input features and the target from the previous trial, were mapped to a 64-dimensional embedding space and positional encoded using sine and cosine functions of different frequencies as in Vaswani et al.¹⁵. Then a causal attention mask was generated for the inputs so that the model makes conditional predictions on all preceding data points. The inputs along the attention mask are then passed to the transformer decoder model, which has six layers, a model dimension of 64, 256 hidden units in the feed-forward network, and eight attention heads. The output of the transformer was then passed through a linear readout and sigmoid function to generate probability estimates for category 1. In practice, inference for all time steps is performed in parallel by passing a causal attention mask to the transformer decoder module in PyTorch⁷⁹. We used binary cross-entropy (BCE) loss for a given batch of inputs and updated the model parameters using the ADAM optimizer⁸⁰ with a learning rate of 10^{-4} . We trained all our models for a total of 500,000 episodes.

Baseline models

Apart from models derived by meta-learning on hand-crafted priors, we considered four other cognitive models as baselines in the domain of category learning, as detailed below.

Rational model of categorization (RMC): The RMC is a Bayesian model of human category learning developed by Anderson et al.⁴³. To derive this model, we simulated data from underlying generative model, such that it followed the data-generating distribution described in Badham et al.⁴², and meta-learned on the generated data, similar to meta-learning on hand-crafted priors. The architecture and training of the model followed the protocol used for ERMI, MI and PFN. We set the free parameters for the RMC based on an earlier study⁴⁰ to the following values: $c = 0.318$, $s_P = 0.488$, and $s_L = 0.046$. However, we did not account for these parameters in our model comparisons, which could explain why the predictive performance RMC is overestimated.

Prototype-based model (PM): Over the years, many different versions of the prototype model have been produced^{38,77}. We used the version from Smith et al.³⁸. This model assigns a category to an observed stimulus based on the similarity distance to the prototype from each category. Specifically, the similarity distance between the stimulus and a prototype, q_k , for category k is calculated as a weighted sum of absolute differences in the dimensions of the features n , with $w_j \in [0, 1]$ corresponding to the weights per feature. The weights are normalized to sum up to 1 as shown in Equation 6.

$$d_{x,q_k} = \sum_{j=1}^n w_j |x_j - q_{k,j}|, \quad (6)$$

The prototypes themselves can be learned or directly specified during model definition. In our case, we assume the prototypes for the two categories $\{q_1, q_2\}$ as a learnable parameter and learn them during the model fitting procedure. That is, $q_{k,j} \in [0, 1] \forall j = \{1, 2, \dots, n\}$ are assumed to be learnable model parameters. The similarity distance between prototypes and stimuli is converted into a psychological space using:

$$\eta_{x,q_k} = e^{-c \cdot d_{x,q_k}} \quad (7)$$

where c is a sensitivity parameter that can shrink or amplify discriminability in a psychological space. The probability of the stimulus being assigned to the category $k = 1$ was then calculated using the following.

$$P(k = 1 | x) = \frac{\eta_{x,q_1}}{\eta_{x,q_1} + \eta_{x,q_2}} \quad (8)$$

Furthermore, the predicted likelihood of the final model is a mixture between the predicted probability of the model and a random guess, with the guessing parameter ε controlling the mixture probabilities.

$$p(k = 1 | x) = (1 - \varepsilon)P(k = 1 | x) + \varepsilon \cdot K^{-1} \quad (9)$$

where K indicates the number of categories.

Generalized context model (GCM): GCM is an exemplar-based model of human category learning developed by Nosofsky et al.⁴⁵. The GCM assigns an observed stimulus to a category by comparing the sum of its similarity scores to all previously seen exemplars in each category, $\{C_1, C_2\}$. The raw distance between the observed stimulus and the exemplars and the similarity score were calculated based on Equations 6 and 7, respectively. The posterior probability of category membership $k = 1$ is calculated on the basis of normalized similarity scores as follows.

$$P(k = 1 | x) = \frac{\sum_{y \in C_1} \eta_{x,y}}{\sum_{y \in C_1} \eta_{x,y} + \sum_{y \in C_2} \eta_{x,y}} \quad (10)$$

The final likelihood of category membership is computed as a mixture between the estimate of posterior probability and a random guessing model as mentioned in Equation 9.

Rule: The rule model considered as the baseline in this work assigns a stimulus to a category based on one of the two rules, whichever better explains the choices of the participants. The first rule is based on the values taken by stimulus features along one dimension, and the second is based on the application of the conjunctive rule on pairs of features, whether a given pair of stimulus features takes on the same value. The final category membership is determined by a mixture between the predicted posterior class probabilities of the model and a random guess, as discussed in Equation 9.

Model fitting

The parameters of all models in the domain of category learning were fit to human data using maximum likelihood estimation. We explain the exact implementation details for the different model classes in the following. The complete list of the parameters fitted for each model is shown in Table S2.

MI, PFN, RMC and ERMI: For models derived using meta-learning, we fitted the inverse temperature term β within the sigmoid function, which squashes the output from the final layer of the transformer to be within $[0, 1]$, to each participant. This term was set to a value of 1 during meta-learning to allow us to derive a Bayes-optimal model and was only fitted during the evaluation phase (bounded to be within $[0, 10]$), with the rest of the model weights frozen. For parameter fitting, we used the differential evolution optimizer available in the SCIPY optimization library⁶⁷.

GCM and PM: We fit the three parameters common to GCM and PM, namely feature weights, sensitivity, and the random guessing parameter, with feature weights bounded to lie within the range $[0, 1]$ and summing to 1; sensitivity term bounded to lie within the $[0, 20]$ range; and the guessing parameter bounded to be within $[0, 1]$. The prototype model also required learning the prototypical stimulus for each category, which is of the same dimensionality as the input stimulus, with the feature values bounded within $[0, 1]$. For parameter fitting, we used the MINIMIZE module available in the SCIPY optimization library.

Rule: We used the same procedure as above except that we learn the stimulus dimension v_i on which the rule is applied.

CLAUDE-V2: We used the same procedure as above except that only the guessing parameter, ϵ , is learned.

Table S2. This table provides the complete list of model parameters that were fit to human data in the domain of category learning, where β is the inverse temperature term, w_i indicates the weights for the stimulus feature dimension i , n is the number of stimulus feature dimensions, c is the sensitivity term, ϵ is noise term in an epsilon greedy policy, q_1 and q_2 are the values for the prototypes for d stimulus features, and v_i are the stimulus dimension on which the rule is applied.

MODEL	PARAMETERS
ERMI, MI, PFN, RMC	β
GCM	$c, \epsilon, w_i \quad \forall i \in \{1, 2, \dots, n\}$
PM	$c, \epsilon, w_i, q_{1,i}, q_{2,i} \quad \forall i \in \{1, 2, \dots, n\}$
RULE	v_1, v_2, ϵ
CLAUDE-V2	ϵ

Bayesian model comparison

After fitting the model parameters to human data using maximum likelihood estimation, we computed the Bayesian information criterion (BIC), which penalizes model fitting performance based on its complexity, for models m for a given participant as follows:

$$\text{BIC}_m = -2 \cdot \max_{\theta_m} \sum_{t=1}^T \log p_{\theta_m}(\hat{y}_t | x_{1:t}, y_{1:t-1}) + |\theta_m| \log(T) \quad (11)$$

where $|\theta_m|$ is the number of parameters estimated for the model m , T is the number of trials in the task and $\hat{y}_t$ is the choice made by the participant in a given trial t .

Once computed, we compared the goodness-of-fit between models using posterior model frequency, which measures how often a given model offers the best explanation in the population. For computing it, we used a Python implementation of the Variational Bayesian Analysis (VBA) toolbox⁷⁴. The toolbox required providing log-evidences for each model and participant pair, which we approximate using $-0.5 \cdot \text{BIC}_m$; see Rigoux et al. study⁷³ for details about this model comparison procedure.

Table S3. Mean performance of humans and models for each rule type in replication of³⁷ study over 15 blocks. Human data was taken from Table 1 in⁴⁰.

Model	Rule						MSE
	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	
Humans	.0201	.0565	.1015	.1120	.1212	.2048	.0000
ERMI	.0586	.0891	.0855	.0826	.0888	.1172	.0287
MI	.0686	.4089	.2404	.1431	.2880	.4201	.2627
PFN	.0170	.3405	.1533	.0226	.2371	.3975	.1736
RMC	.1329	.2215	.1903	.1718	.2132	.3364	.1003

CLAUDE-V2 as a cognitive model of human category learning

To simulate the study by Badham et al.⁴² using CLAUDE-V2, we queried the model with the prompt shown below. Geometric stimuli from the original experiment were described in text format. The order of presentation of the stimulus was randomized and the number of presentations per block was compared to the original study. As the Claude API returns only sampled tokens,

not log-probabilities, we coded predictions as binary outcomes, $\pi(k = 1 \mid x_t; x_{1:t-1}, y_{1:t-1})$. The final model predicted category probabilities is again a mixture between the category prediction from the model and a random guess as mentioned in Equation 12. We conducted 96 simulation runs for each of the six categorization rules.

$$p(k = 1 \mid x_t) = (1 - \varepsilon)\pi(k = 1 \mid x_t; x_{1:t-1}, y_{1:t-1}) + \varepsilon \cdot K^{-1} \quad (12)$$

Prompt for Badham et al. 2017 study

In this experiment, you will be shown examples of geometric objects. Each object has three different features: size, color, and shape. Your job is to learn a rule based on the object features that allows you to tell whether each example belongs in the {A} or {B} category. As you are shown each example, you will be asked to make a category judgment and then you will receive feedback. At first you will have to guess, but you will gain experience as you go along. Try your best to gain mastery of the {A} and {B} categories.

- In trial 1, you picked category {A} for Big Black Square and category {A} was correct.
- In trial 2, you picked category {A} for Small Black Triangle and category {B} was correct

Human: What category would a Small Black Triangle belong to? (Give the answer in the form “Category <your answer>”).
Assistant: Category

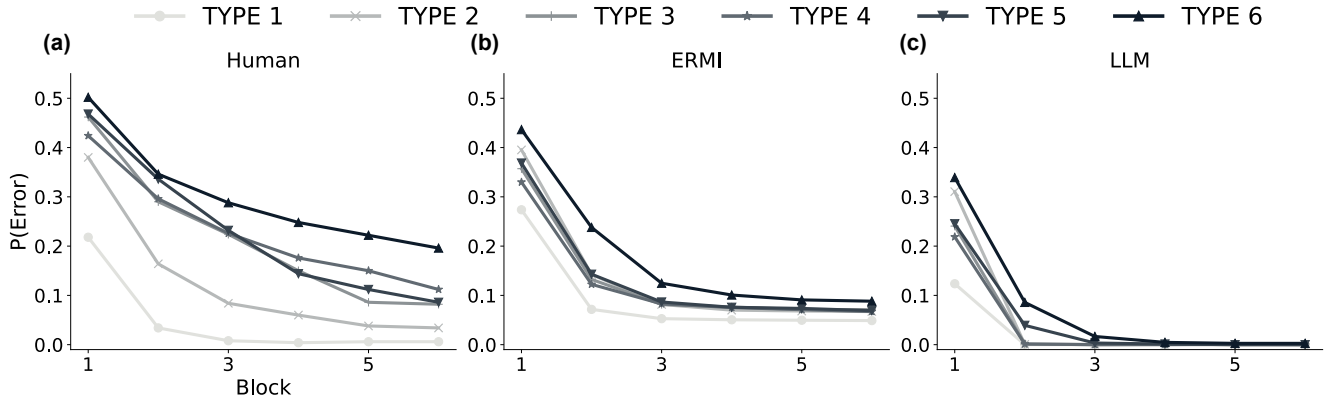


Figure S5. Unlike ERMI, CLAUDE-v2 does not show human-like learning difficulties: (a-c) Average error probabilities for each task *type* in each block of 16 trials for (a) humans, (b) ERMI, and (c) LLM. Human data in (a) was reproduced from Table 1 in Nosofsky et al.⁴⁰ study. ERMI was simulated on *type 1-6* tasks for 50 runs with the inverse temperature set to $\beta = 0.4$. CLAUDE-v2 was simulated for 94 runs each on *type 1-6* tasks with temperature term set to 0.

Decision Making

LLM-generated tasks

Prompts: In the following, we provide the prompts used in the two stages of decision making learning domain. We used the following prompt to synthesize the names of stimulus features and targets, similar to function learning, separately for each of the three conditions, ranking, direction, and unknown.

Synthesize stimulus feature name and its target for ranking condition

I am a psychologist who wants to run a function learning experiment. In a function learning experiment, a real-world feature is mapped to its corresponding target, with both feature and target taking on continuous values.

Please generate names for features and its corresponding target for **250** different function learning experiments. Additionally, order the feature names according to how well they predict the target:

– feature name, target name

Synthesize stimulus feature name and its target for direction condition

I am a psychologist who wants to run a function learning experiment. In a function learning experiment, a real-world feature is mapped to its corresponding target, with both feature and target taking on continuous values.

Please generate names for features and its corresponding target for **250** different function learning experiments. Additionally, the features should be such that higher feature values lead to higher target values:

– feature name, target name

Synthesize stimulus feature name and its target for unknown condition

I am a psychologist who wants to run a function learning experiment. In a function learning experiment, a real-world feature is mapped to its corresponding target, with both feature and target taking on continuous values.

Please generate names for features and its corresponding target for **250** different function learning experiments:

– feature name, target name

Next, we prompted the LLM to generate values for tasks generated from stage 1:

Generate values for ranking condition

I am a psychologist who wants to run a function learning experiment. For a function learning experiment, I need a list of features with their corresponding target. The features in this case are feature1, feature2, feature3, and feature4. These features take on only numerical values and must be continuous. The target, <target>, should be predictable from the feature values and must also have continuous values. Note that the features are listed according to how well each of them can predict the target. The first feature is most useful for predicting the target, the second feature is the second most useful, etc.

Please generate a list of <num-data> feature-target pairs sequentially using the following template for each row: - feature value 1, feature value 2, feature value 3, feature value 4, target value

Generate values for direction condition

I am a psychologist who wants to run a function learning experiment. For a function learning experiment, I need a list of features with their corresponding target. The features in this case are feature1, feature2, feature3, and feature4. These features take on only numerical values and must be continuous. The target, <target>, should be predictable from the feature values and must also have continuous values. Note that the values taken by the features should be such that higher feature values lead to higher target values.

Please generate a list of <num-data> feature-target pairs sequentially using the following template for each row: - feature value 1, feature value 2, feature value 3, feature value 4, target value

Generate values for unknown condition

I am a psychologist who wants to run a function learning experiment. For a function learning experiment, I need a list of features with their corresponding target. The features in this case are feature1, feature2, feature3, and feature4. These features take on only numerical values and must be continuous. The target, <target>, should be predictable from the feature values and must also have continuous values.

Please generate a list of <num-data> feature-target pairs sequentially using the following template for each row: - feature value 1, feature value 2, feature value 3, feature value 4, target value

Parsing and pre-processing: The parsing expressions used and the data preprocessing steps are the same as in the function learning domain.

Qualitative analysis of synthesized input features and labels: We show the counts for the top-50 most frequently occurring names for (a) input features and (b) targets in Figure S6. We found that the LLM tends to produce input-target pairs that are relevant to everyday life such as supply-demand influence on productivity, diet-genetics influence on weight change, cloud cover-humidity on crop yield, study time-intelligence quotient on test score, etc.

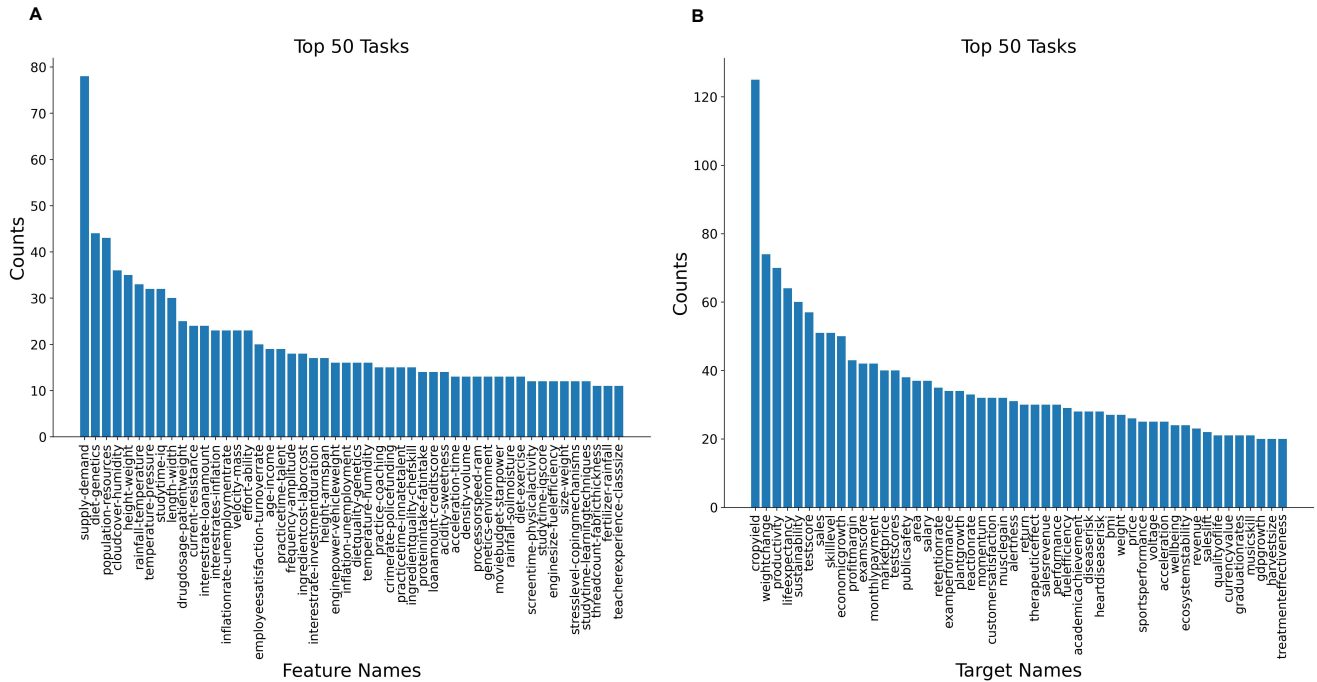


Figure S6. Frequency of input and target labels in CLAUDE-V2 synthesized decision making tasks: Counts for the top-50 most frequently occurring (a) two-dimensional input feature names and (b) target names computed over 9254 LLM-generated decision learning tasks belonging to the unknown condition. These distributions confirm that the LLM-generates real-world functional relationships that are useful for everyday decision making.

Data-distributional properties of LLM-generated tasks: We generate three datasets of decision-making tasks, one for each of (A) unknown, (B) ranking, and (C) direction, following the prompts described above. To examine their properties and verify if the manipulation was successful, we computed four key statistics: input correlations, sparsity in predictive features, ranking of feature importance, and directionality of features with respect to the target, and compared them across datasets. Specifically, we contrasted the ranking and direction conditions with the unknown condition, which served as a baseline. We found that the first feature was more often the most important feature in terms of predictive power (see the caption of Figure S7 for details on the calculation) in the ranking condition (51.76%) than in the unknown condition (43.75%). Likewise, the proportion of features positively correlated with the target was higher in the direction condition (92.46%) than in the unknown condition (79%).

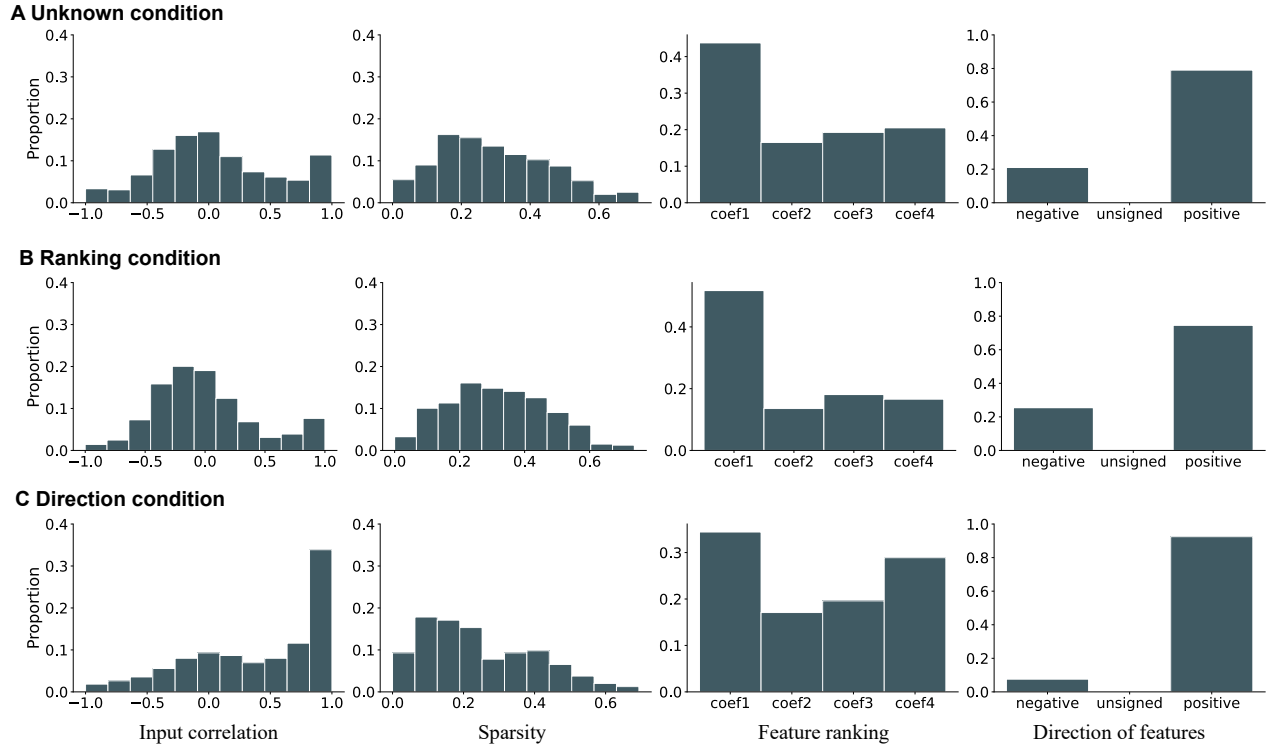


Figure S7. Data-distributional properties of LLM-generated decision making tasks for (A) unknown, (B) ranking and (C) direction condition. Histogram of Pearson correlation coefficients between all distinct pairs of normalized input features (first column). Histogram of Gini coefficients computed on the absolute ordinary least squares (OLS) weights when regressing the normalized target on all normalized inputs with an intercept (second column; higher values indicate sparser weights). Histogram of the index of the input feature with the largest absolute per-feature OLS weight, where each per-feature model regresses the target on a single feature with an intercept (third column; feature ranking). Histogram of the sign of per-feature OLS weights from those single-feature-with-intercept regressions (fourth column; direction).

Human studies

Binz et al. 2022⁴⁹. This study was conducted on 27 participants in total, with each participant performing 30 different paired comparison tasks. Tasks were generated by first sampling feature weights from a standard normal distribution. Feature vectors for each option were then drawn from a multivariate normal distribution with zero mean and fixed covariance. Finally, the binary choice outcome was determined by sampling from a Bernoulli distribution, where the success probability was given by a probit regression over the difference in feature values (see Equation 2 in the main paper). The feature weights were kept the same within a task, which consisted of 10 trials, but were resampled between tasks. All participants performed the same set of paired comparison tasks but presented in randomized order. In Experiment 3a, participants observed two features per option, whereas in Experiment 3b they observed four features per option. In neither of these two experiments, information about the ranking of the features and their directions were provided. The experiment itself was framed as an alien sports competition on an unknown planet. Participants observed two or four numerical attributes for two aliens, depending on the experiment

they were part of. They indicated their choice by pressing a button corresponding to the alien they believed would most likely win. This cover story was used so that the meaning of the feature attributes remained abstract for each participant. Participants were not told about the underlying feature weights, and they had to learn them through trial and error, using the feedback about correct choice provided after each trial. All participants in the experiment performed a short tutorial and went through a comprehension check, which ensured clear understanding of the experimental protocol before data-collection.

Hand-crafted tasks

Synthetic paired-comparison tasks used for training MI: We generated three synthetic datasets of paired-comparison problems (between 7000-9000 tasks per set) under *ranking*, *direction* and *unknown* conditions. For each task, a weight vector $w \in \mathbb{R}^d$ was sampled from a standard normal distribution. In the *direction* condition, weights were constrained to be non-negative by taking absolute values; in the *ranking* condition, feature importance was rank-ordered by sorting weights by magnitude; and in the *unknown* condition, weights were left unconstrained. To generate options, the feature vectors were sampled from a zero-mean multivariate normal distribution with covariance $\Sigma = L \text{diag}(\theta) L^\top$, where L was drawn from an LKJ (Lewandowski–Kuworicka–Joe distribution; $\eta = 2$) prior and $\theta = \mathbf{1}$. The LKJ distribution is a flexible prior over correlation matrices that allows control over the strength of correlations while ensuring positive definiteness. Each trial presented a pair of options $x_a, x_b \sim \mathcal{N}(0, \Sigma)$, with the comparison input defined as $x = x_a - x_b$. We randomly determine which option has the highest criterion by sampling from a Bernoulli distribution as follows: $y \sim \text{Bernoulli}(\Phi(w^\top x / \sigma))$ with $\sigma = 0.1$. Each task contained a maximum of 10 trials, which corresponded to the length of the experiment in which this model was evaluated.

Model architecture, and training

The input vector for a given trial in a decision making task was the difference between the input features for the two options, computed for each dimension independently, and the correct target option from the previous trial. The number of features in the decision making task was either two or four dimensions and the total number of observations in a given task was 20. These inputs were embedded into a 64-dimensional space, with positional encoding applied using sine and cosine functions of varying frequencies, following Vaswani et al.¹⁵. A causal attention mask ensured that predictions at each time step were conditioned only on all previous inputs. These masked sequences were processed using a Transformer decoder composed of six layers, with 64-dimensional embeddings, eight attention heads, and 256 hidden units in the feedforward layers. The decoder outputs were passed through a linear projection to produce weights for the different feature dimensions. The likelihood of a target option is then calculated by first projecting the output through a linear layer, multiplying it element-wise with the current input features, summing across dimensions, and passing the result through a sigmoid to obtain a Bernoulli probability. Training was performed using the negative log-likelihood (NLL) loss over all input observations in a batch. The model parameters were updated as mentioned before using the SCHEDULEFREE optimizer⁷¹ with a baseline learning rate of 3×10^{-4} . Each model was trained for 100000 episodes, with periodic evaluation on held-out tasks to monitor generalization performance.

Baseline models

Apart from the MI model derived by meta-learning on tasks generated with hand-crafted priors, we considered three other cognitive models as baselines in the domain of decision making, as detailed below.

Single-cue decision maker (SC): In Equation 13, we demonstrate formally how the heuristic of single-cue decision making makes a decision given the input feature. Note that x^* indicates that the model only takes into account a single feature, which in this case was the most predictive feature. This means that only one parameter is fitted to human choices.

$$p(y_t = 1 \mid x_t, \theta_m, m = \text{SC}) = \Phi\left(\frac{\theta_m \cdot x_t^*}{\sqrt{2}\sigma}\right) \quad (13)$$

where Φ is the cumulative distribution function of a standard normal distribution, θ_m is the weight of the selected feature, and σ is the noise standard deviation.

Equal weighting decision maker (EW): We considered a probabilistic version of the equal weighting model, as shown in Equation 14. When $w > 0$, this model probabilistically selects the option with the larger sum of features. In contrast, when $w < 0$, it selects the option with the smaller sum of features. Once again, only one parameter is fitted to the human data.

$$p(y_t = 1 \mid x_t, \theta_m, m = \text{SC}) = \Phi\left(\frac{\theta_m \cdot \sum_{i=1}^d x_{t,i}}{\sqrt{2}\sigma}\right) \quad (14)$$

where Φ is the cumulative distribution function of a standard normal distribution, θ_m is the feature weight, and σ is the noise standard deviation.

Feedforward neural network (NN): We used a feedforward neural network from the Binz et al.⁴⁹ study as an additional baseline model. This model predicts the target given the difference between the input features of the two options and the previous target as input. The network consisted of a single hidden layer with 128 units followed by two linear transformations projected to the mean and (log) standard deviation of a normal distribution. The neural network parameters were trained by gradient descent on the negative log-likelihoods of the target. During model fitting, the learning rate parameter and the inverse temperature term were fit to human choices; see Appendix F in Binz et al.⁴⁹ for implementation details.

Model fitting and comparison

For fitting the model parameters, we performed the maximum likelihood estimation using Bayesian optimization⁸¹, following the procedure used by Binz and colleagues.⁴⁹ A complete list of model parameters that are fitted to human choices can be found in Table S4. Upon fitting, we followed the same exact steps as described above for category learning for Bayesian model comparison. That is, we used a VBA tool box, where we provide $-0.5 \cdot \text{BIC}_m$ as an approximation of log-evidence for each model and participant; see Rigoux et al. study⁷³ for details.

Table S4. This table provides the complete list of model parameters that were fit to human data in the domain of category learning, where β is the inverse temperature term, θ indicates the weights for the stimulus feature dimension, and α is learning rate.

MODEL	PARAMETERS
ERMI, MI	β
SC	θ
EQ	θ
NN	α, β

Alternative LLMs

During the early stages of this work, we also considered two other LLMs: Llama-2⁸² and GPT-4⁸³, which were among the best performing models at the time. However, the non-instruction-tuned Llama-2 (the only version available at the time) could not consistently produce the 100+ data points required for each category learning task. Its outputs were also difficult to parse, as they frequently failed to follow the specified format. More recently, with Llama-3.1 (70B)⁸⁴, we were able to generate decision-making datasets whose quality matched those produced by CLAUDE-V2.

Preliminary analysis with GPT-4 revealed that it often sampled input features from a uniform distribution, relying on its internal coding module. It also tended to generate only simple heuristic rules, such as requiring the sum of two features to exceed the third, or the mean of two features to be greater than another, for assigning an input to its category. Furthermore, statistical analysis on a small GPT-4 generated dataset showed that its task statistics closely resembled those of category learning tasks with hand-crafted priors (specifically Bayesian logistic regression prior). Due to this lack of diversity in the generated task statistics, we decided to use CLAUDE-V2 over GPT-4.

References

1. Darwin, C. *On the Origin of Species* (John Murray, London, 1859).
2. Brunswik, E. *Perception and the Representative Design of Psychological Experiments* (University of California Press, Berkeley, 1956).
3. Simon, H. A. Rational choice and the structure of the environment. *Psychol. Rev.* **63**, 129–138, DOI: [10.1037/h0042769](https://doi.org/10.1037/h0042769) (1956).
4. Gigerenzer, G., Todd, P. M. & the ABC Research Group. *Simple Heuristics That Make Us Smart* (Oxford University Press, New York, 1999).
5. Anderson, J. R. *The Adaptive Character of Thought* (Lawrence Erlbaum Associates, Hillsdale, NJ, 1990).
6. Goldstein, D. G. & Gigerenzer, G. Models of ecological rationality: The recognition heuristic. *Psychol. Rev.* **109**, 75–90, DOI: [10.1037/0033-295X.109.1.75](https://doi.org/10.1037/0033-295X.109.1.75) (2002).
7. Brown, T. B. *et al.* Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020).

8. Borisov, V., Seßler, K., Leemann, T., Pawelczyk, M. & Kasneci, G. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280* **arXiv** (2022).
9. Zhu, J.-Q. & Griffiths, T. L. Eliciting the priors of large language models using iterated in-context learning. *arXiv preprint arXiv:2406.01860* **arXiv** (2024).
10. Jagadish, A. K., Coda-Forno, J., Thalmann, M., Schulz, E. & Binz, M. Human-like category learning by injecting ecological priors from large language models into neural networks. In *Forty-first International Conference on Machine Learning* (2024).
11. Marewski, J. N., Gaissmaier, W. & Gigerenzer, G. Good judgments do not require complex cognition. *Cogn. Process.* **10**, 117–128, DOI: [10.1007/s10339-009-0264-y](https://doi.org/10.1007/s10339-009-0264-y) (2009).
12. Ortega, P. A. *et al.* Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030* **arXiv** (2019).
13. Binz, M. *et al.* Meta-learned models of cognition. *Behav. Brain Sci.* **47**, e147 (2024).
14. Bischl, B. *et al.* Openml benchmarking suites. *arXiv:1708.03731v2 [stat.ML]* **arXiv** (2019).
15. Vaswani, A. *et al.* Attention is all you need. *Adv. neural information processing systems* **30** (2017).
16. Hochreiter, S., Younger, A. S. & Conwell, P. R. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, 87–94 (Springer, 2001).
17. Wang, J. X. *et al.* Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* **arXiv** (2016).
18. Lake, B. M. & Baroni, M. Human-like systematic generalization through a meta-learning neural network. *Nature* **Nature**, 1–7 (2023).
19. Carroll, J. D. Functional learning: The learning of continuous functional mappings relating stimulus and response continua. *ETS Res. Bull. Ser.* **1963**, i–144 (1963).
20. Koh, K. & Meyer, D. E. Function learning: Induction of continuous stimulus–response relations. *J. Exp. Psychol. Learn. Mem. Cogn.* **17**, 811–836, DOI: [10.1037/0278-7393.17.5.811](https://doi.org/10.1037/0278-7393.17.5.811) (1991).
21. Brehmer, B. Hypothesis testing, probability learning, and a simple rule. *J. Exp. Psychol.* **102**, 887–891, DOI: [10.1037/h0036211](https://doi.org/10.1037/h0036211) (1974).
22. DeLosh, E. L., Bussemeyer, J. R. & McDaniel, M. A. Extrapolation: the sine qua non for abstraction in function learning. *J. Exp. Psychol. Learn. Mem. Cogn.* **23**, 968 (1997).
23. Schulz, E., Tenenbaum, J. B., Duvenaud, D., Speekenbrink, M. & Gershman, S. J. Compositional inductive biases in function learning. *Cogn. psychology* **99**, 44–79 (2017).
24. Schulz, E., Tenenbaum, J. B., Reshef, D. N., Speekenbrink, M. & Gershman, S. J. Assessing the perceived predictability of functions. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 37 (2015).
25. Lichtenberg, J. M. & Şimşek, Ö. Simple regression models. In *Imperfect decision makers: Admitting real-world rationality*, 13–25 (PMLR, 2017).
26. Bussemeyer, J. R., Byun, E., DeLosh, E. L. & McDaniel, M. A. Learning functional relations based on experience with input–output pairs by humans and artificial neural networks. In *Knowledge concepts and categories*, 405–437 (Psychology Press, 2013).
27. Kwantes, P. J. & Neal, A. Why people underestimate y when extrapolating in linear functions. *J. Exp. Psychol. Learn. Mem. Cogn.* **32**, 1019 (2006).
28. Kalish, M. L., Lewandowsky, S. & Kruschke, J. K. Population of linear experts: knowledge partitioning and function learning. *Psychol. review* **111**, 1072 (2004).
29. Little, D. R., Shiffrin, R. M. & Laham, S. M. Function estimation: Quantifying individual differences of hand-drawn functions. *Mem. & Cogn.* **Springer**, 1–20 (2024).
30. Brehmer, B. Hypotheses about relations between scaled variables in the learning of probabilistic inference tasks. *Organ. Behav. Hum. Perform.* **11**, 1–27 (1974).
31. Brehmer, B. Single-cue probability learning as a function of the sign and magnitude of the correlation between cue and criterion. *Organ. Behav. Hum. Perform.* **9**, 377–395 (1973).
32. Brehmer, B., Kuynlenstierna, J. & Liljergren, J.-E. Effects of function form and cue validity on the subjects’ hypotheses in probabilistic inference tasks. *Organ. Behav. Hum. Perform.* **11**, 338–354 (1974).

33. Byun, E. *Interaction between prior knowledge and type of nonlinear relationship on function learning*. Ph.D. thesis, Purdue University (1995).
34. McDaniel, M. A. & Busemeyer, J. R. The conceptual basis of function learning and extrapolation: Comparison of rule-based and associative-based models. *Psychon. bulletin & review* **12**, 24–42 (2005).
35. Lucas, C. G., Griffiths, T. L., Williams, J. J. & Kalish, M. L. A rational model of function learning. *Psychon. bulletin & review* **22**, 1193–1215 (2015).
36. Ashby, F. G. & Maddox, W. T. Human Category Learning. *Annu. Rev. Psychol.* **56**, 149–178, DOI: [10.1146/annurev.psych.56.091103.070217](https://doi.org/10.1146/annurev.psych.56.091103.070217) (2005).
37. Shepard, R. N., Hovland, C. I. & Jenkins, H. M. Learning and memorization of classifications. *Psychol. Monogr. Gen. Appl.* **75**, 1–42, DOI: [10.1037/h0093825](https://doi.org/10.1037/h0093825) (1961).
38. Smith, J. D. & Minda, J. P. Prototypes in the mist: The early epochs of category learning. *J. Exp. Psychol. Learn. memory, cognition* **24**, 1411 (1998).
39. Johansen, M. K. & Palmeri, T. J. Are there representational shifts during category learning? *Cogn. Psychol.* **45**, 482–553, DOI: [10.1016/s0010-0285\(02\)00505-4](https://doi.org/10.1016/s0010-0285(02)00505-4) (2002).
40. Nosofsky, R. M., Gluck, M. A., Palmeri, T. J., McKinley, S. C. & Glauthier, P. Comparing models of rule-based classification learning: a replication and extension of shepard, hovland, and jenkins (1961). *Mem. Cogn.* **22**, 352–369, DOI: [10.3758/bf03200862](https://doi.org/10.3758/bf03200862) (1994).
41. Devraj, A., Zhang, Q. & Griffiths, T. The dynamics of exemplar and prototype representations depend on environmental statistics. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 43 (2021).
42. Badham, S. P., Sanborn, A. N. & Maylor, E. A. Deficits in category learning in older adults: Rule-based versus clustering accounts. *Psychol. Aging* **32**, 473–488, DOI: [10.1037/pag0000183](https://doi.org/10.1037/pag0000183) (2017).
43. Anderson, J. R. The adaptive nature of human categorization. *Psychol. Rev.* **98**, 409–429, DOI: [10.1037/0033-295X.98.3.409](https://doi.org/10.1037/0033-295X.98.3.409) (1991).
44. Homa, D. & Cultice, J. C. Role of feedback, category size, and stimulus distortion on the acquisition and utilization of ill-defined categories. *J. Exp. Psychol. Learn. Mem. Cogn.* **10**, 83 (1984).
45. Nosofsky, R. M. Attention, similarity, and the identification–categorization relationship. *J. experimental psychology: Gen.* **115**, 39 (1986).
46. Müller, S., Hollmann, N., Arango, S. P., Grabocka, J. & Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510 arXiv* (2021).
47. Samuels, R., Stich, S. & Bishop, M. Ending the rationality wars. *Collect. Pap. Vol. 2: Knowledge, Ration. Morality, 1978-2010* **2**, 191 (2012).
48. Camerer, C., Loewenstein, G. & Prelec, D. Neuroeconomics: How neuroscience can inform economics. *J. economic Lit.* **43**, 9–64 (2005).
49. Binz, M., Gershman, S. J., Schulz, E. & Endres, D. Heuristics from bounded meta-learned inference. *Psychol. review Psychological review* (2022a).
50. Geisler, W. S. Sequential ideal-observer analysis of visual discriminations. *Psychol. review* **96**, 267 (1989).
51. Gigerenzer, G. & Gaissmaier, W. Heuristic decision making. *Annu. review psychology* **62**, 451–482 (2011).
52. Chater, N. & Vitányi, P. Simplicity: a unifying principle in cognitive science? *Trends cognitive sciences* **7**, 19–22 (2003).
53. Todd, P. M. & Gigerenzer, G. Environments that make us smart: Ecological rationality. *Curr. directions psychological science* **16**, 167–171 (2007).
54. Martignon, L. & Hoffrage, U. Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory Decis.* **52**, 29–71 (2002).
55. Goldstein, D. G. & Gigerenzer, G. The recognition heuristic: How ignorance makes us smart. In *Simple heuristics that make us smart*, 37–58 (Oxford University Press, 1999).
56. Farrell, H., Gopnik, A., Shalizi, C. & Evans, J. Large ai models are cultural and social technologies. *Science* **387**, 1153–1156 (2025).
57. Tenenbaum. Joshua Tenenbaum’s homepage. <http://web.mit.edu/cocosci/josh.html> (2021). [Online; accessed 9-November-2021].

58. Todd, P. M. & Brighton, H. Building the theory of ecological rationality. *Minds Mach.* **26**, 9–30, DOI: [10.1007/s11023-015-9371-0](https://doi.org/10.1007/s11023-015-9371-0) (2016).
59. Jagadish, A. K., Binz, M., Saanum, T., Wang, J. X. & Schulz, E. Zero-shot compositional reinforcement learning in humans. *PsyArXiv preprint PsyArXiv:ymve5* **PsyArXiv** (2023).
60. Jagadish, A. K., Coda-Forno, J., Thalmann, M., Binz, M. & Schulz, E. Bounded ecologically rational meta-learned inference explains human category learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 47 (2025).
61. Peterson, J. C., Bourgin, D. D., Agrawal, M., Reichman, D. & Griffiths, T. L. Using large-scale experiments and machine learning to discover theories of human decision-making. *Science* **372**, 1209–1214 (2021).
62. Ji-An, L., Benna, M. K. & Mattar, M. G. Automatic discovery of cognitive strategies with tiny recurrent neural networks. *bioRxiv* **bioRxiv**, 2023–04 (2023).
63. Miller, K., Eckstein, M., Botvinick, M. & Kurth-Nelson, Z. Cognitive model discovery via disentangled rnns. *Adv. Neural Inf. Process. Syst.* **36**, 61377–61394 (2023).
64. Bauer, J. *et al.* Human-timescale adaptation in an open-ended task space. *PMLR PMLR*, 1887–1935 (2023).
65. Anthropic, P. B. C. Claude 2. <https://www.anthropic.com/index/claude-2> (2023). Accessed: 2024-1-15.
66. Pedregosa, F. *et al.* Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
67. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **17**, 261–272, DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2) (2020).
68. Chan, S. *et al.* Data distributional properties drive emergent in-context learning in transformers. *Adv. Neural Inf. Process. Syst.* **35**, 18878–18891 (2022).
69. Anderson, J. R. Is human cognition adaptive? *Behav. brain sciences* **14**, 471–485 (1991).
70. Schubert, J. A., Jagadish, A. K., Binz, M. & Schulz, E. In-context learning agents are asymmetric belief updaters. In *International Conference on Machine Learning*, 43928–43946 (PMLR, 2024).
71. Defazio, A. *et al.* The road less scheduled. *Adv. Neural Inf. Process. Syst.* **37**, 9974–10007 (2024).
72. Müller, S., Hollmann, N., Arango, S. P., Grabocka, J. & Hutter, F. Transformers can do bayesian inference. In *International Conference on Learning Representations* (2022).
73. Rigoux, L., Stephan, K. E., Friston, K. J. & Daunizeau, J. Bayesian model selection for group studies—revisited. *Neuroimage* **84**, 971–985 (2014).
74. Daunizeau, J., Adam, V. & Rigoux, L. Vba: a probabilistic treatment of nonlinear models for neurobiological and behavioural data. *PLoS computational biology* **10**, e1003441 (2014).
75. Binz, M. & Schulz, E. Modeling human exploration through resource-rational reinforcement learning. *Adv. Neural Inf. Process. Syst.* **35**, 31755–31768 (2022b).
76. Tummars, B. Datathief iii <http://datathief.org>. *Datathief is a program used to reverse engineer data points from a graph* (2006).
77. Medin, D. L. & Schaffer, M. M. Context theory of classification learning. *Psychol. Rev.* **85**, 207–238, DOI: [10.1037/0033-295x.85.3.207](https://doi.org/10.1037/0033-295x.85.3.207) (1978).
78. Nosofsky, R. M., Palmeri, T. J. & McKinley, S. C. Rule-plus-exception model of classification learning. *Psychol. Rev.* **101**, 53–79, DOI: [10.1037/0033-295x.101.1.53](https://doi.org/10.1037/0033-295x.101.1.53) (1994).
79. Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, 8024–8035 (Curran Associates, Inc., 2019).
80. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
81. Team, T. G. Gpyopt: A bayesian optimization framework in python. <http://github.com/SheffieldML/GPyOpt> (2016).
82. Touvron, H. *et al.* Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
83. Achiam, J. *et al.* Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
84. Meta Platforms, I. Llama 3.1 70b model (2024).