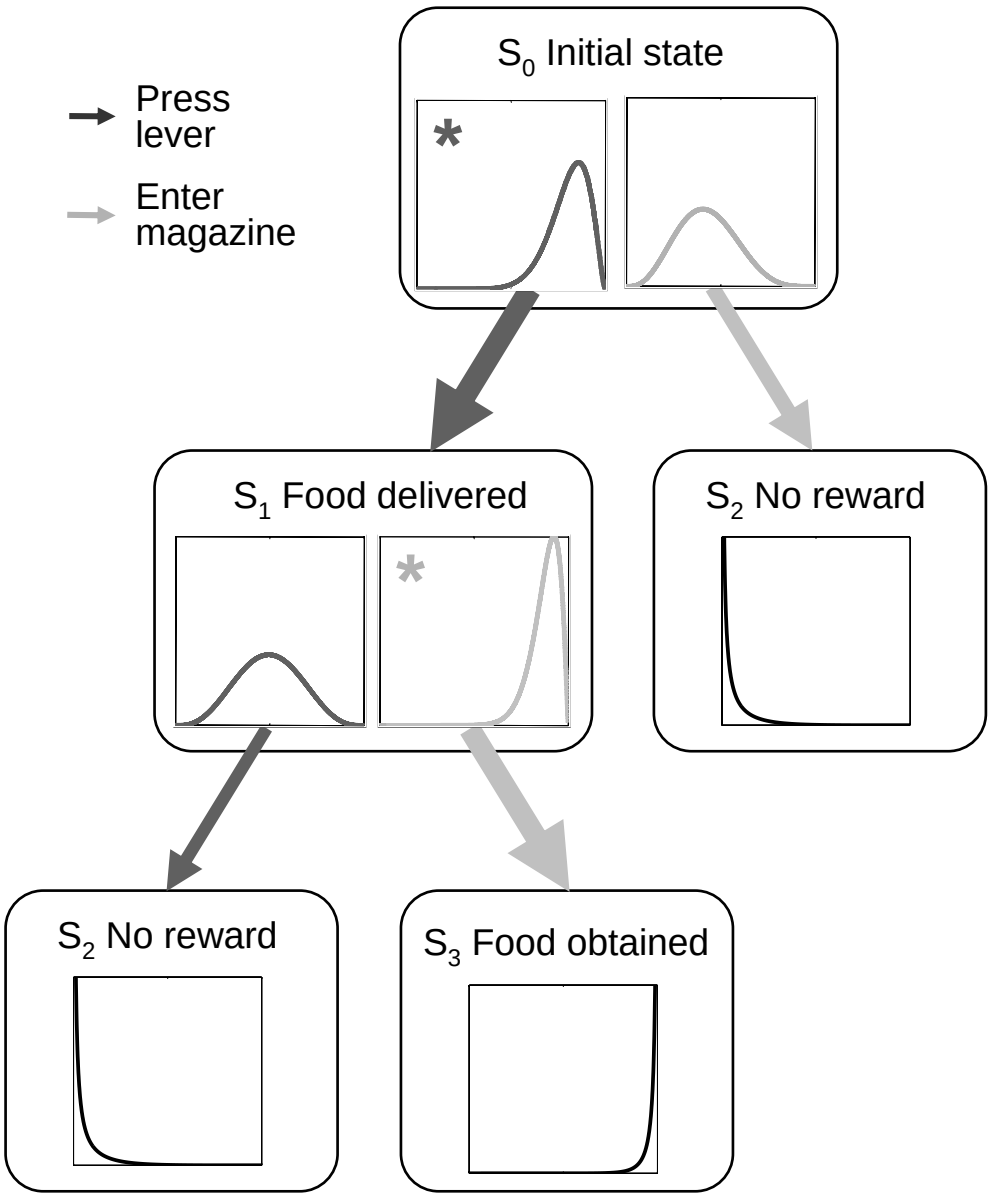
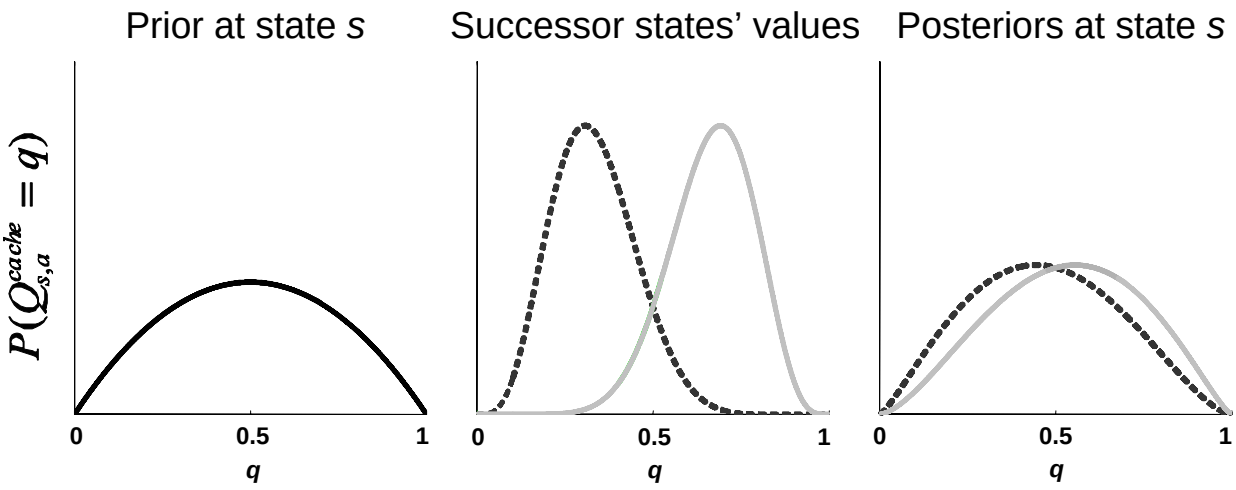


Supplementary Figure 1

Value propagation in tree search, after 50 steps of learning the task in **Figure 1a**. The inset plots show the distributions over state-action values, $\mathbf{Q}^{\text{tree}}_{s,a}$, computed by the tree system from the learned distributions over the values of the terminal states (shown in black) and over the transition structure of the task. The distributions are plotted as the probability assigned to each possible value q. Their moments are given by iteration on Equations 5 and 6 in **Supplementary Methods** — each value distribution is a function of the value distributions for the best action (marked with an asterisk) at each possible successor state. Arrows represent the most likely transition for each state and action, and their widths are proportional to the likelihoods (the full set of mean transition probabilities is illustrated in **Fig. 4**). The better actions were better explored and hence more certain (narrower value distributions); distributions at each state were similar to the distributions at the most likely successor state, and more so when transition to that state was more likely. As iterations progressed backwards, distributions got broader.



Supplementary Figure 2

Example of learning in the cache algorithm, following a single transition from state s to s' having taken action a. The leftmost panel shows the prior distribution $\mathbf{Q}^{\text{cache}}_{s,a}$. The middle panel plots the distribution over the value of the successor state s' (specifically, the distribution $\mathbf{Q}^{\text{cache}}_{s',a'}$ for the best action a' in s'). The curves illustrate two different successors, a more and less favorable state. For both of the successor states, the right panel plots the posterior distribution (whose moments are given by Equations 3 and 4 in **Supplementary Methods**) over the original state and action, $\mathbf{Q}^{\text{cache}}_{s,a}$, as updated following the visit to s'. The effect of learning was to nudge the predecessor state's value distribution in the direction of the value of the successor state.

Supplementary methods

Here, we describe our implementation of uncertainty-tracking in the caching and tree-search systems, and then report the parameters governing these models and comment on the effect of slow changes in the task.

Recall that the goal of each system is to estimate, for each state s and action a, a (factorial) distribution $\mathbf{Q}_{s,a}$ over the future expected value $Q(s,a)$. The latter is defined:

$$Q(s,a) \equiv \begin{cases} R(s) & s \text{ is terminal } (a = \varnothing) \\ \sum_{s'} T(s,a,s') \cdot \max_{a'}[Q(s',a')] & \text{otherwise} \end{cases} \tag{1}$$

where $\varnothing$ stands in for the fact that no action is possible at a terminal state. Tree and cache systems use different approximations for value estimation, and we use the uncertainty as a measure of how appropriate these methods are in particular circumstances. However, the uncertainty computation itself requires a number of further shortcuts. In our implementation, these were matched between the systems so far as possible in the hope that even though the absolute values of the uncertainty measurements are likely erroneous, the errors should be similar between systems and thus their relative uncertainties should be informative about the actual reliabilities of the underlying value estimation methods.

Caching algorithm

We learned $\mathbf{Q}^{\text{cache}}$ from experience using "Bayesian Q-learning"[1] (for a different Bayesian formulation see Engel et al.[2]), adapted to our simplified class of MDPs. Notionally, the

method involves assuming prior distributions $\mathbf{Q}^{\text{cache}}_{s,a}$ describing uncertainty about the value of each state and action, and then updating them using Bayes' theorem to reflect subsequent experience. We made four simplifying assumptions: that the distributions $\mathbf{Q}^{\text{cache}}_{s,a}$ were at each step expressed as beta distributions $Beta(\alpha_{s,a}, \beta_{s,a})$; that the distribution $\mathbf{Q}^{\text{cache}}_{s,a}$ was independent of $\mathbf{Q}^{\text{cache}}_{s',a'}$ for $s \neq s'$ or $a \neq a'$; that the distribution of the maximum (with respect to action choice) of a state-action value was just the distribution of values, $\mathbf{Q}^{\text{cache}}_{s,a}$, for the action a optimizing the mean $\langle \mathbf{Q}^{\text{cache}}_{s,a} \rangle$; and that the bootstrapped posterior distribution for a nonterminal state was specified by Dearden et al.'s[1] "mixture update" approximation. See Dearden et al.[1] for a full discussion of the merits of these simplifications; the most serious is the assumption that different states' values were independent, contrary to the coupling inherent in their definition (Equation 1). Our assumptions differ from Dearden's mainly in our use of a beta distribution for the posterior. We evaluated this last simplification against the same method implemented with an arbitrary posterior (finely discretized for numerical estimation), and found it innocuous. (In particular, over 500 steps of our task, the largest deviation between the methods' variance or mean estimates at any state was 0.5%). We nonetheless stress that the numerical accuracy of the approximate Bayesian computations is not key to our argument — we intended rather to implement both caching and tree-search learning using similar approximations, so that any computational biases impacted both similarly.

For the details of the method, consider being in state s — either s is a terminal state and we receive reward $r \in \{0, 1\}$, or we take action a and transition to another state s'. For

terminal states s with prior $Beta(\alpha_{s,\varnothing}, \beta_{s,\varnothing})$ and reward r, Bayes' theorem specifies that the posterior value distribution $\mathbf{Q}^{\text{cache}}_{s,\varnothing}$ will be distributed as $Beta(\alpha_{s,\varnothing}+r, \beta_{s,\varnothing}+(1-r))$. For nonterminal states s followed by s', we wish to treat the successor state's mean value $\langle \mathbf{Q}^{\text{cache}}_{s',a'} \rangle$ (for the action a' optimizing that mean) as a bootstrapped sample of the predecessor state's mean value; the question is how to take into account uncertainty about the two states' values. If the successor state's value were $0 \leq q \leq 1$ we might, by analogy with the terminal state case, take the predecessor state's value posterior $\mathbf{Q}^{\text{cache}}_{s,a}$ as $Beta(\alpha_{s,a}+q, \beta_{s,a}+(1-q))$. Following Dearden et al.[1], we used the mixture of such distributions with respect to the successor state value distribution $\mathbf{Q}^{\text{cache}}_{s',a'}$:

$$\int_0^1 Beta(\alpha_{s,a}+q, \beta_{s,a}+(1-q)) \mathbf{Q}^{\text{cache}}_{s',a'}(q) dq \tag{2}$$

Though this integral is neither readily solvable nor itself a beta distribution, its mean and variance are analytic, and we thus approximated the predecessor state's posterior value $\mathbf{Q}^{\text{cache}}_{s,a}$ as the beta distribution matching those moments. They are:

$$\mu^{\text{cache}}_{s,a} = \frac{\alpha_{s,a} + \langle q \rangle^{\text{cache}}_{s',a'}}{n_{s,a}+1} \tag{3}$$

$$(\sigma^2)^{\text{cache}}_{s,a} = \frac{1}{(n_{s,a}+2)(n_{s,a}+1)} (\alpha^2_{s,a} + \alpha_{s,a} + \langle q^2 \rangle^{\text{cache}}_{s',a'} + (2\alpha_{s,a}+1)\langle q \rangle^{\text{cache}}_{s',a'}) - (\mu^2)^{\text{cache}}_{s,a} \tag{4}$$

where $\langle q \rangle^{\text{cache}}_{s',a'}$ and $\langle q^2 \rangle^{\text{cache}}_{s',a'}$ are respectively the first and second moments of the beta distribution $\mathbf{Q}^{\text{cache}}_{s',a'}$ and $n_{s,a} = \alpha_{s,a} + \beta_{s,a}$.

To model outcome devaluation (e.g., treatments in which the animal is allowed to

sample the outcome in the devalued state, but not in the context of the task), we replaced the distribution $\mathbf{Q}_{s,\varnothing}^{\text{cache}}$ for the terminal state s corresponding to the devalued outcome, reducing its expected value. Absent further learning with samples of trajectories ending in this state, this has no effect on the cached values of any other states.

Tree-search algorithm

A Bayesian tree-search method[3–5] involves two stages: model identification, and value computation. To identify the MDP, we assumed beta priors over the reward functions and, for each state and action, a Dirichlet prior over the vector of successor state probabilities. For these simulations, we assumed the number of states and their terminal or nonterminal status were known. As experience accrues, these distributions can be updated exactly by Bayes' theorem, simply by counting state transitions and rewards.

Given posterior distributions over the state transition and reward functions, a standard "certainty equivalent" method for estimating expected values $Q(s, a)$ would be to assume the true MDP is described by the means of those distributions and then to solve for the values using *value iteration*, or repeated application of Equation 1[6]. This is roughly equivalent to using tree search to compute the values of all states in parallel. Here we wish to quantify the uncertainty about the values that results from the uncertainty about the transition and reward functions. An optimal (if impractical) way to do so would be to repeat the value iteration process for all possible combinations of transition and reward functions, weighting each resulting set of state-action values by the probability of

the transition and reward functions that produced it. Such an approach can be directly approximated by sampling from the distribution over MDPs[3,4]. Here, we used a set of approximations more closely matched to the Bayesian Q-learning methods discussed above — by performing not a set of iterations over future value for different trees, but rather a single iteration on the *distributions* over the future values implied by the distributions over the trees[5]. We assumed, as before, that at all search steps k the distributions $\mathbf{Q}_{s,a}^{\text{tree},k}$ were expressed as beta distributions; that these distributions were, at each step, independent of one another for different states or actions, as were the posterior distributions over transition and reward functions; and that the distribution of the maximum (with respect to action choice) of a state-action value was the distribution corresponding to the single, apparently optimal action. See Mannor et al.[5] for analysis and experiments on the accuracy of a similar approach.

In particular, we initialized the 0-step value distribution $\mathbf{Q}_{s,a}^{\text{tree},0}, \forall(s,a)$ as equal to the beta distribution over reward probability $R(s)$. We did this for both terminal and nonterminal states — the immediate reward distribution for nonterminal states was determined from the same prior by conditioning on the absence of reward each time the state is visited since in our simplified MDPs, reward is only available at terminal states. Then, for nonterminal states s, we repeatedly searched a further step down the tree, estimating the k-step value distributions $\mathbf{Q}_{s,a}^{\text{tree},k}$ as a function of the $k-1$-step value distributions $\mathbf{Q}_{s,a}^{\text{tree},k-1}$. As before, a $\mathbf{Q}$ distribution was approximated by the beta distribution matching the mean and variance of the (complicated) exact distribution. These are just the moments of Equa-

tion 1 (which describes the probability of future reward if the transition and successor state value functions were known) with respect to the distributions over those functions. After action a in state s, the probabilities $t_1 \dots t_n$ of transitioning to states $s_1 \dots s_n$ are Dirichlet-distributed, and the successor states' values $q_1 \dots q_n$ are beta-distributed (each state's as $\mathbf{Q}^{\text{tree},k-1}_{s_i,a_i}$ for the apparently best action a_i). Taking into account our independence assumptions, the mean and variance of $\mathbf{Q}^{\text{tree},k}_{s,a}$ are:

$$\mu^{\text{tree},k}_{s,a} = \sum_{i=1}^{n} \langle t_i \rangle \langle q_i \rangle \tag{5}$$

$$(\sigma^2)^{\text{tree},k}_{s,a} = \sum_{i,j:i \neq j} \langle t_i t_j \rangle \langle q_i \rangle \langle q_j \rangle + \sum_{i=1}^{n} \langle t_i^2 \rangle \langle q_i^2 \rangle - (\mu^2)^{\text{tree},k}_{s,a} \tag{6}$$

where the bracketed values are standard Dirichlet and beta moments for the distribution over T and the successor state $\mathbf{Q}$ distribution. This iteration was repeated until the distributions converged.

In more realistic domains with many states, it is impractical to re-compute value distributions at every state. This can be addressed by a number of methods, including pruning (examining only certain paths out of each state at each step). We modeled the "computational noise" or inaccuracy that would result as extra variance accumulating over each step of tree search. In particular, at each step k, we added a penalty to the variance $(\sigma^2)^{\text{tree},k}_{s,a}$ of a constant ν times the probability that the successor state s' was nonterminal.

To model reward devaluation, we replaced the distribution over $R(s)$, the reward probability for the terminal state corresponding to the devalued outcome, reducing its

expected value. Through the value iteration, this immediately impacted all subsequently computed estimates $\mathbf{Q}^{\text{tree}}_{s,a}$.

Noise model, priors, and parameters

The final complexity is that we assumed a nonstationary task — that is, that the MDP functions T and R could change randomly over time. Rather than employing an explicit, generative model of change, we captured nonstationarity using an exponential forgetting heuristic, whereby at each timestep, the parameters defining the cache system's distributions $\mathbf{Q}^{\text{cache}}$ and the tree system's distributions over transition and reward functions decayed exponentially (with factor γ) toward their respective priors at each timestep. Such decay captures the decline in relevance of past samples given possible intervening change. As the decay factors were matched between controllers (as were the priors), this corresponds to equivalent time horizons on past data — i.e., equivalent assumptions about the speed of change of the MDP.

While the qualitative effects we demonstrated are robust, our theory has a number of free parameters. One advantage of a normative approach is that these are often not arbitrary quantities (like a "learning rate") but rather assertions about regularities in the external environment, such as how quickly tasks change. Thus, although they are at present chosen rather arbitrarily, they suggest directions for future experimental test.

Parameters for our simulations were as follows. The softmax parameter β was 5. The tree system's prior over the transition functions was a symmetric Dirichlet with parame-

ter $\alpha = 1.0$ and over the reward functions was $Beta(0.1, 0.1)$ (encoding a prior assumption that outcome utilities are likely deterministic). The cache system's priors over the Q functions were matched: $Beta(0.1, 0.1)$ for terminal states, and for nonterminal states the same beta distribution implied by tree search on the tree system's prior over MDPs. The step penalty ν was 0.005, and the reward distribution for a devalued outcome in both tree and cache systems was $Beta(1, 15)$. The exponential forgetting factor γ was 0.98.

Simulations were run 250-1,000 times and means reported (results vary between runs due to stochastic action choice). In all cases, confidence intervals on the plotted quantities (s.e.m.) were too small to visualize; error bars were thus omitted.

References

1. Dearden, R., Friedman, N. & Russell, S.J. Bayesian Q-learning. in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)* 761–768 (1998).

2. Engel, Y., Mannor, S. & Meir, R. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. in *Proceedings of the 20th International Conference on Machine Learning (ICML)* 154–161 (2003).

3. Dearden, R., Friedman, N. & Andre, D. Model based Bayesian exploration. in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)* 150–159 (1999).

4. Strens, M. A Bayesian framework for reinforcement learning. in *Proceedings of the 17th International Conference on Machine Learning (ICML)* 943–950 (2000).

5. Mannor, S., Simester, D., Sun, P. & Tsitsiklis, J.N. Bias and variance in value function estimation. in *Proceedings of the 21st International Conference on Machine Learning (ICML)* 568–575 (2004).

6. Sutton, R.S. & Barto, A.G. *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).